1988

The creation of machine and software educational environments appropriate to the cognitive processing characteristics of children

Raymond Stace
University of Wollongong, rstace@uow.edu.au

THE CREATION OF MACHINE AND SOFTWARE EDUCATIONAL ENVIRONMENTS APPROPRIATE TO THE COGNITIVE PROCESSING CHARACTERISTICS OF CHILDREN

A thesis submitted in fulfilment of the requirements for the award of the degree of

MASTER of EDUCATION (Honours)

from

THE UNIVERSITY OF WOLLONGONG

by

RAYMOND STACE, B. A., Grad. Dip. Ed. Stud. (Comp. in Ed.)

EDUCATION

1988

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

This study provides an examination of the difficulties faced by educators when confronted by the new computer technology. An examination of Computer Assisted Learning provides an understanding of the types of uses to which a computer can be put in an educational environment. Software is divided into four basic types: instructional, revelatory, emancipatory, and conjectural. This section includes a discussion of the strengths and weaknesses of the different types of software.

An examination is made of theories of learning in so far as they contribute to the design of educational software. In particular, the contribution of theorists such as Skinner, Gagné and Bandura is looked at, and models are provided for the development of three different types of software: drill and practice, tutorial, and simulation. These models provide educators with an appreciation of the design approach to be taken in the development of such software, as well as with a starting point should they wish to design software themselves.

Theories of intellectual development are examined with the aim of assisting the educator to match the performance and the needs of each student with the available software. The contributions of Piaget, Case, Bruner, and Gagné are looked at and an overview is provided on information processing, attention and memory, concept organisation, language, cognitive mapping, and social cognition.

Methods of evaluating such software are also discussed in order that the educator may be better able to determine whether or not a particular piece of software is suitable for students in a particular curriculum context. Particular attention is paid to software descriptions and requirements as well as to the development of a software evaluation checklist.

Detailed evaluations are then conducted of two quite different pieces of educational software. The packages are evaluated on the program description, the content, the program features, any special features of the program, the classroom application, and the support materials provided.

The software design problem is next examined by partitioning the topic into five main areas of design. These are: clarity, simplicity, attractiveness and motivational appeal, efficiency, and support materials. Finally, the design in detail of a sample piece of software and its subsequent field evaluation is examined with the aim of providing the educator with a deeper insight into the overall demands of the software design problem.

For his patience, encouragement, good humour and wisdom in guiding me towards a deeper understanding of the subject of this thesis, I am extremely grateful to my supervisor Professor Ron King.

I
INTRODUCTION

At the same time as the computer comes to play an ever increasing role in the educational environment, education itself is likely to play a much more significant role in human endeavour. Writers such as Tom Stonier (1984) have already attempted to capture some aspects of this vision of the future. The pursuit of such a vision, however, is dependent upon great change both in society and in education. "All this [change] will take time and prolonged effort. The struggle will reach into every corner of society. But that is precisely what is at stake: a new universal accessibility." (Williams, 1974, 151) This 'accessibility' is accessibility to information through the new computer technology. Part of this struggle will be the need for agreement to be reached on the best way to utilise the technology, as "... the introduction of technology on a wide scale eventually depends on a social consensus." (Reinecke, 1982, 241)

The widespread use of computer technology will allow society to redefine the concepts of school, work, and employment (Jones, 1982). Stonier sees a future in which most formal learning will be done at home using computers, while schools will serve the purpose of providing socialising games and activities. If this is to happen, the role of the teacher will change, and may become more like the ancient Greek teacher, tutor or guiding hand, leading students through an education. Part of this new role for the teacher will be to act as an interface between the child and the terminal and the real world outside. According to Franks, this will mean arranging and providing the child with real world experiences, such as excursions, sports, arts, craft, music, drama, and so on. "The teacher's role is shifting, and

will continue to shift from omniscient mentor to information guide." (Franks, 1984, 42)

The question arises as to whether or not this is merely another shift in the teacher's role similar to those which have been occurring throughout the history of mass formal education. The cycle from teacher-centred to child-centred and back again has been repeated many times, and especially so during the last twenty years. Some of these have been the emergence of the 'Open' school, the use of coloured rods, reading in colour, school based curriculum development, the various 'process' methods of teaching subjects, and so on. From the practitioner's point of view, many of these seemed to appear almost overnight and most disappeared just as quickly. Often they were very localised – down to the classroom level at times, so that you might have one class using coloured rods, while the one next to it was not – and just as often there was no support for the practitioner in the implementation of these changes. Often, the teacher was simply unable to find the time to implement the changes in a professional manner.

In personal discussions with King, the question was raised as to whether there was anything different about the use of computers in education which would ensure that this shift was not a transient one. One major difference about computers as a social innovation would seem to be the universality of their use in education (not merely at the school level) and in so many facets of our daily lives and occupations. As King (1985) himself states:

> We are facing, in formal education, what is potentially the most substantial and influential innovation in education since the introduction of universal schooling. The skeletal structure of that innovation is already with us in the classroom in the form of the microchip, the computers that the microchip has spawned, the software currently arriving on the education market, and the already well-tried satellite- and line-based communications technology that is all around us.

This change towards a greater use of computers is not something which is only being implemented at the class, school, district, state, or even national level. It is a

change that is occurring internationally at all levels of education and of society. It is not a change that is being driven by any individual teacher, theorist or by any group, (except, perhaps, by computer manufacturers for entirely different reasons) but is being driven by the community at large, which is concerned that its children are educated in the 'use' of computers. Thus, we are seeing an increase in the use of computers in education. A second major difference with this innovation is the power available through the computer in relation to learning. As Papert states:

> ...my conjecture is that the computer can concretize (and personalize) the formal. Seen in this light, it is not just another powerful educational tool. It is unique in providing us with the means for addressing what Piaget and many others see as the obstacle which is overcome in the passage from child to adult thinking. I believe that it can allow us to shift the boundary separating concrete and formal. Knowledge that was accessible only through formal processes can now be approached concretely. And the real magic comes from the fact that this knowledge includes those elements one needs to become a formal thinker. (Papert, 1980, 21)

The ever increasing use of the computer in education will more easily allow education to be seen as a life-long concern which people can take up or leave at any time of life. This would include University courses conducted via computer, allowing people to make use of the new technology to study for degrees and higher qualifications, or to train or re-train for different jobs, all done from the home. Such study would be open to all without qualification, with progression being dependent upon satisfactory work being completed. Such a system could provide equal opportunities for all.

The computer's role in these processes will be a multi-faceted one. The computer will be able to provide access to an electronic data base which will hold more information than any library of today. What is more, this information will be immediately available to all. The computer will not only be able to provide individual tuition for students, but it will provide interactive tuition. Because learning will be a more private activity than it is today, more positive attitudes towards all aspects of learning can develop. "Instead of the passively defeatist:

'I'm not good at this', the child would say: 'How can I make myself better at it?'." (Boden, 1980, 445) Students will not be so bothered by peer pressure, as when they are seen to be obviously (and, perhaps, stupidly) wrong before the whole class. Recent research has in fact found that when compared to traditional instruction, Computer Assisted Learning "... tended to yield lower scores [better results] on various aspects of mathematics anxiety — an outcome that may have important practical implications for school [and, indeed, for all] learning." (Mevarech and Ben-Artzi, 1987, 46)

The computer will help facilitate a step-by-step approach to learning, with much stress being placed upon readiness. For students who study programming, the use of the computer will help to develop an organised intellectual approach to problem solving. By developing critical thinking powers and by providing a multitude of simulations of real life situations, students can be trained to think laterally in the application of these powers. Through computers it becomes possible to present material in much more interesting and varied ways, especially if those computers are linked to interactive video disc programs, for example. Visual or pictorial information is much more easily assimilated than spoken or written information.

Computers "... open up new possibilities for teaching and learning at all levels." (Mulock, 1983, 2) As a result, "... students, teachers, administrators and parents need to become aware of how computers can be used to increase the effectiveness of learning and in the management of information and resources." (Mulock, 1983, 4) Since the computer is merely a tool, educators need to gain an understanding of Computer Assisted Learning (CAL) in order to make it an

effective and efficient tool. To gain such an understanding educators need:

1. to learn about the different software types, what they are used for, and their strengths and weaknesses (to be examined in chapter two of this study);
2. to learn about the ways in which CAL works (to be examined in chapter three of this study);
3. to learn about the cognitive processing characteristics of children (to be examined in chapter four of this study);
4. to be able to evaluate the merits and demerits of CAL software in relation to their own specific curriculum needs (to be examined in chapters five and six of this study); and
5. to develop an appreciation for the CAL software design problem (to be examined in chapters seven, eight and nine of this study).

As part of the examination of the Computer Assisted Learning software design problem, this study will examine the development of a software package and will also evaluate that software package through trials in schools. The aim of this evaluation will be to test the effectiveness of the design of the learning environment created by the program. It is not intended in this study to evaluate the effectiveness of that environment in assisting children to investigate and to learn number concepts.

II
COMPUTER ASSISTED LEARNING

There are many uses of the computer in an educational environment. These uses include Computer Assisted Learning (CAL) or Instruction (CAI)† and Computer Managed Learning (CML). This work is concerned with CAL, and discovering what types of CAL are available, and how they can be used. In general, the range of different possibilities for CAL can be depicted as in Figure 2·1. At the lowest end of the spectrum, the computer is seen by many to be used as an unsophisticated teaching machine only (teacher centred). In this environment the computer is in total control of the student using the program. At the highest end of the spectrum, the computer is being used as an extremely sophisticated learning device. In this environment, the student is seen by many to have full control over the computer (student centred).

The matter of the student being in control or otherwise is, however, a difficult question in the CAL context. When using a basic tutorial program, the computer may seem to be in control. If the student has the ability to choose for himself or herself the tasks to be done when using the program does this mean that the student is in control? Is the computer really in control on the other hand, because there is only a controlled number and kind of task available to the student when using the program? At the opposite end of the computer environmental scale as shown in Figure 2·1, when the student is programming a computer to carry out a particular task is that student really in control of the computer or is the computer in control, because the environment is limited by the available commands and functions

† In this paper CAI = CAL. The term CAL is used throughout.

Types of Educational program†

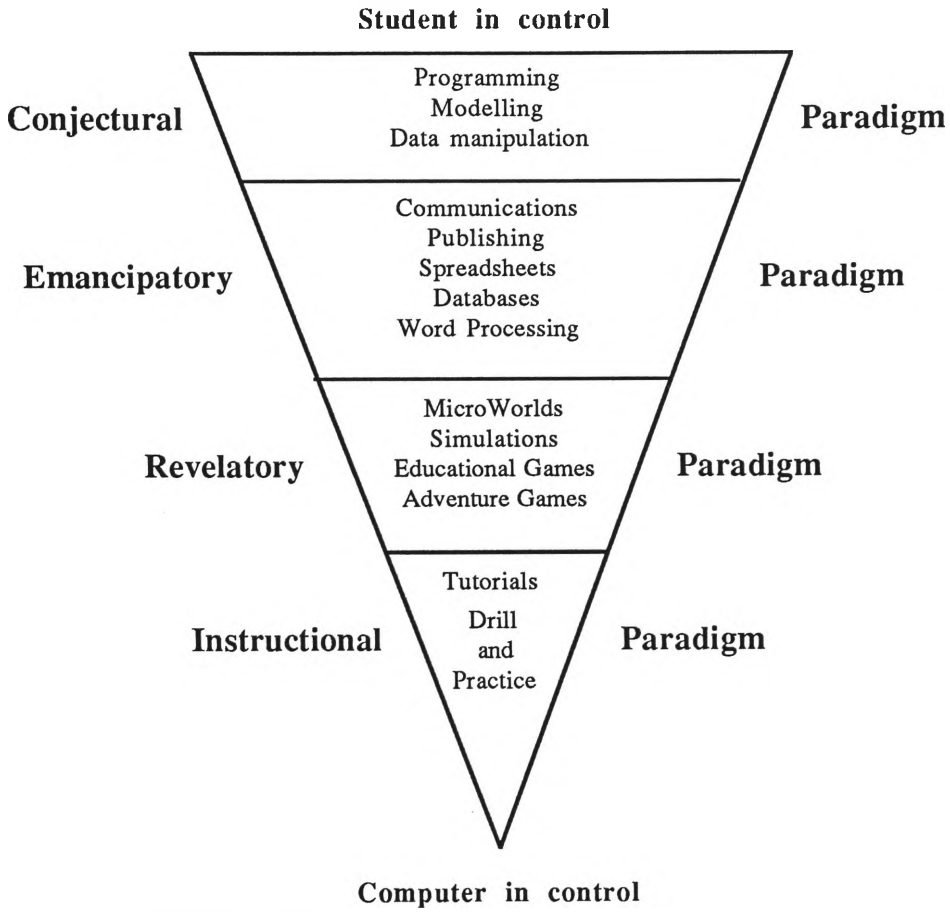


Figure 2·1 † Adapted from Wellington, 1985

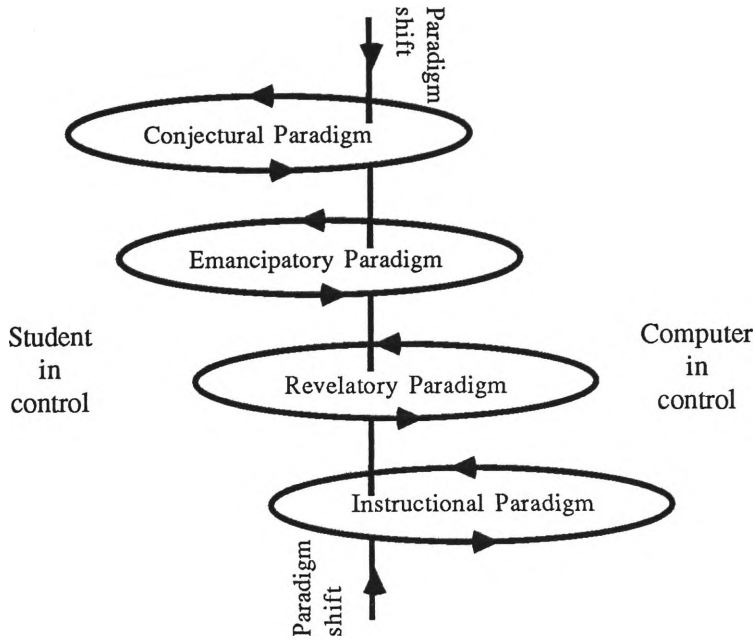


Figure 2·2 Paradigms of program control

The term 'complexity' is here used in relation to the knowledge and skills content being presented to the learner within each of the program paradigms. In applying this term to the software paradigms being examined, this means that the cognitive level of skills and knowledge placed before the user of instructional software is the least complex, while that placed before the user of conjectural software is the most complex. An example from the instructional software paradigm could involve the relatively simple skill of adding one number to another and arriving at a total, while using software from the conjectural paradigm could involve the student in applying much more complex cognitive skills in tasks such as mathematical, economic, or societal modelling and analysis.

The term 'complexity' is not being used here as it has been in the computing science field by Kearney† and others as a measurement of resources used and of computer-program-programmer interaction.

---

† Kearney, J.K., Sedlmeyer, R.L. et al. 1986. Software complexity measurement. *Communications of the ACM*. Volume 29, Number 11.

defined within that programming language? Perhaps the task the student would like to have the computer carry out simply cannot be done because of the programming language limitations, although it may be able to be done if another programming language was used. With these questions in mind, it may be in fact that one example of a given software program type (tutorial software, for example) may be teacher centred while another example of the same software type may be student centred. The working relationship between the teacher, the student, and the software designer (through the computer program itself) and the variety of ways in which a particular program might be used means that any one program could be seen to fit within more than one CAL paradigm. Such an outcome could be represented as in Figure 2·2. The *Number Detective* program (Greig & Stace, 1985e) for example, could be used by the teacher to drill number counting. It is, however, a microworld, designed to be used in an exploration and discovery mode by the student. The software designers have no control over the use of the package once it has reached the classroom.

There are a number of different types of CAL. These types of CAL differ in the method of lesson presentation, levels of response or interaction offered, and complexity. The different types of CAL have been delineated within four paradigms by Rushby (1979). These paradigms are:

— the instructional;

— the revelatory;

— the emancipatory; and

— the conjectural.

Each of these will be examined in turn.

THE INSTRUCTIONAL PARADIGM

The aim of this program type is to teach the learner a given piece of subject matter or to impart a specific skill. It involves breaking a learning task into a series of sub-tasks, each with its own stated prerequisites and objectives. These separate tasks are then structured and sequenced to form a coherent whole. This programmed learning approach was initially based upon Skinner's theory of conditioning. This theory proposed that subject material be broken down into small tasks which were presented one at a time. When a task was mastered a reward was given to reinforce the learning, and the next task in the sequence was tackled. Software conforming to the instructional paradigm generally falls into one of two types: drill and practice software or tutorial (programmed instruction) software.

Instructional software designers have generally, however, only made use of Skinner's ideas on breaking the subject matter down into small segments. They have not heeded Skinner's advice that the student should be in control of the learning situation. The reason Skinner proposed that subject matter be reduced to its smallest possible unit, is that he believed that the student would then be more easily able to understand the subject matter and would be able to control learning. The computer program is able to be adapted to provide the option of selecting a beginning difficulty level, of skipping sequences, of exiting at will, and of re-doing a level. CAL tutorials maximise student control by not following a rigid path from the beginning to the end of a lesson and by providing facilities to analyse student errors, provide hints, provide reinforcement, and advise on remedial courses of action if necessary. Although more powerful computers than are generally available in the classroom are necessary to be able to achieve all of these in a single program, there are several excellent tutorial programs, such as *Addition Made Easy, Subtraction Made Easy, Multiplication Made Easy*, and *Long Division Made Easy* (Greig & Stace, 1985) which track student progress and analyse errors in a step-by-

step fashion and provide functional hints as necessary. By making use of such features, this type of instructional software puts the student in greater control than might be realised after looking at Figure 2·1. In Skinner's words:

> The best way to help the student give birth to the answer he is struggling to recall is to give him a strong hint or even the whole answer, but that is not the best way to make sure that he will recall it in the future. Polya is correct in saying that the heuristic hint "Do you know a related problem?" is to be preferred to the stronger hint "Could you apply the theorem of Pythagoras?" The student will solve the present problem more quickly with the stronger hint, but he will learn more about solving problems if the weak hint works. As Comenius said, "The more the teacher teaches, the less the student learns" ... the better the teacher, the more important it is that he frees the student from the need for instructional help. (Skinner, 1968, 144)

Drill and Practice Software

When using this type of software, the student is randomly presented with a structured set of questions or exercises designed to provide practice in a particular skill, or to drill recall of particular knowledge. Slightly more sophisticated drill and practice programs analyse responses and suggest further action. The typical drill and practice program follows a cycle such as the one illustrated in Figure 2·3. The time taken to answer a question may be taken into account when answers are being evaluated. Different levels of difficulty are usually provided for. Drill and practice programs may involve a competitive game between two players or between one player and the computer.

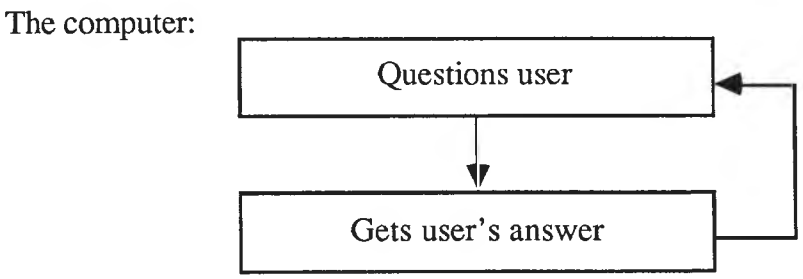


Figure 2·3 Drill and practice software cycle

Tutorial Software

When using this type of software, the student is led through a set of learning material via a structured question and answer dialogue, following a cycle such as the one illustrated in Figure 2·4. After being presented with data, the student is then asked a set of questions on the information given. The student's answers are compared with a set of stored answers, some form of reinforcement is usually given as a response, and the student is moved on. If the answer provided was correct, new work will be presented, but if incorrect, the student may be branched to remedial work on the same topic.

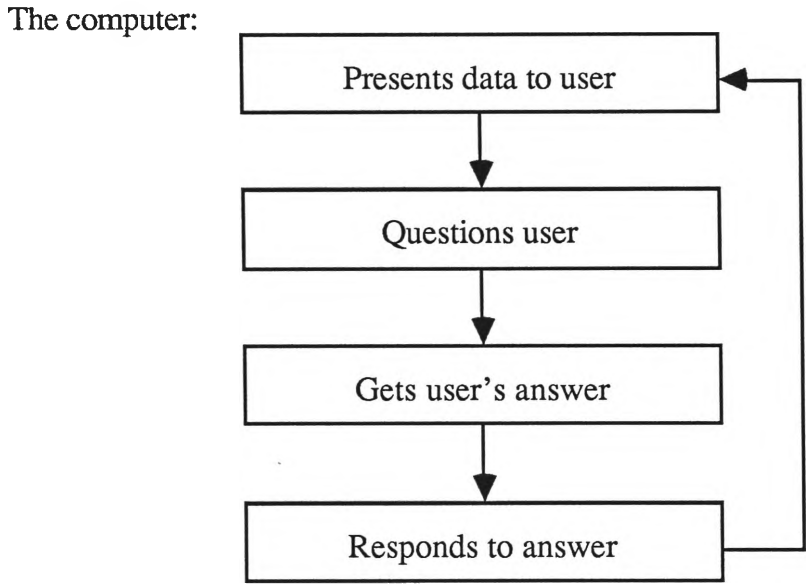


Figure 2·4 Tutorial software cycle

Instructional or 'teaching' programs are usually of the following types:

1. Programs to test, reinforce, or revise a body of knowledge. E.g.: maths tables, foreign language vocabulary, and so on.
2. Programs to test a particular skill. E.g.: addition, spelling.
3. Programs that provide practice in using a particular principle or concept in order to develop an understanding. E.g.: by using a particular

scientific principle over and over again in different circumstances or to develop an understanding of a concept such as area or volume.

4. Tutorial or direct instruction programs. These programs are designed to tutor students, as their name suggests. They often introduce new concepts. Most tutorial programs are set up to introduce or describe a concept or rule, provide examples of same, and have the learner practice the understanding of the concept or the application of the rule by using similar examples. The more sophisticated tutorial programs may also test, revise, and suggest remediation activities.

Instructional Software Strengths

The strengths of instructional software are that:

- It can easily present and teach facts and skills.
- It can easily test or question knowledge of facts and possession of skills.
- It can easily provide drill and practice.
- It can easily provide (limited) reinforcement.
- It can easily provide for individual differences.
- It can be personalised by frequently using the student's name.
- It can be used in a competitive situation if needed. (E.g.: to provide intrinsic motivation.)
- It can provide extrinsic motivation. (E.g.: "Help the monkey reach the coconut!")

Instructional Software Weaknesses

The weaknesses of instructional software are that:

- Not all subject matter can easily be broken up into small parts, each with clearly defined prerequisites and objectives.
- Not all students respond to the behaviourist learning model stimuli in the way that that theory assumes.
- There can be serious problems of controlling progress, checking understanding at each step and of access to the program because of its individualised lesson nature.

THE REVELATORY PARADIGM

The aim of this program type is to guide a student through a process of learning by discovery. The subject matter and its underlying model or theory are progressively revealed to the student as he or she uses the program and develops a feeling for its behaviour under varying circumstances and thus is led to discover the rules of that model or theory. Revelatory or 'learning' programs are usually of the following types.

1. Educational games
2. Adventure games
3. Simulations

   Some examples of simulations are:

   3.1 Lab experiments

   3.2 Historical

   3.3 Business/marketing

   3.4 Geographical

   3.5 Ecological pollution/fuel

4. Problem Solving

5. MicroWorlds (Discovery)

Revelatory Software Strengths

The strengths of revelatory software are that:

- Generally, the student is in control of the computer (within the limitations of the program).
- It can assist in developing the skills of forming and testing hypotheses.

  As Bruner states:

  > ... to the degree that one is able to approach learning as a task of discovering something rather than learning about it, to that degree will there be a tendency for the child to carry out his learning activities with the autonomy of self reward or, more properly by reward that is discovery itself.
  > (Bruner, 1974b, 406)

- It can pose moral and social dilemmas to be solved.
- It can assist in developing skills in the area of problem solving and of logical and sequential thought.
- It can assist in the development of creative thinking skills.
- It can assist in the development of oral language skills.
- It can assist in improving concentration skills (Molloy, 1986).
- Comprehension and reading skills can be developed as students recognise clues, recognise cause and effect relationships and distinguish between fact and fiction.
- Properly constructed games can lead to the formation of strategies and knowledge structures that have general usefulness in other domains (Burton & Brown, 1982). For example, note-taking, map and diagram drawing skills may be developed by some adventure games.

- It can act as a stimulus for other activities, such as writing (Molloy, 1986), the taking of notes and the maintenance of a diary (Dailhou, 1986b).
- The use of simulations can have various advantages, such as:
    - saving money (on chemicals, transport);
    - saving time (setting up experiment, planning a visit);
    - in the simulation of a long time-scale process (statistical projections);
    - safety (dangerous reactions);
    - simulating invisible processes (sub-atomics); and
    - control over outcomes.
- Adventure games allow for role playing and for experimentation with different kinds of behaviour and unusual situations (Dailhou, 1986a). The student has to analyse the situation and act according to the rules which govern the game's environment.
- Where students work together as a group, there is the opportunity for social and interactive skills to develop.

> Games and sport not only are pleasant forms of play but also have great value as socializing agents. From them the child learns how to get along with other children, to cooperate in different activities, to play the role of leader as well as a follower, and to evaluate himself and his abilities realistically by comparing himself with his playmates. (Hurlock, 1964, 462)

This applies equally to computer games and activities.

Revelatory Software Weaknesses

The weaknesses of revelatory software are that:

- The design of some games may include only a very limited range of knowledge or skills to be developed with the result that the game itself

overshadows any educational merit. These should, perhaps, be regarded as recreational software rather than revelatory software.

- It is difficult to prevent the student from forming incorrect models of the underlying structure of the software (Burton & Brown, 1982).
- It is difficult to help the student see the limitations of his/her strategies (Burton & Brown, 1982).
- It is difficult to help the student discover the causes of his/her errors (Burton & Brown, 1982).
- The over-use of simulations may lead to a tendency for teachers to prefer simulations to the real thing (because of the advantages listed above). From the student's point of view, this may lead to unwanted misconceptions about how physical processes and natural phenomena really work, thus providing a distorted vision of reality.
- The over-reliance on simulations may also mean that students do not acquire the skills necessary to conduct experimental studies.
- Each simulation is presenting only a small slice of reality for manipulation. This means that by necessity, not all aspects of reality are presented within the model provided. The student needs to be made aware of this, and to be able then to critically evaluate the model for what it is, and question its limitations and sources.
- There can be a danger of one group monopolising the use of the software and the computer at the expense of others. Recent research in the United Kingdom has found, for instance, that there has been a drop of more than 50 per cent since 1980 in the number of women studying computing at university (*The Australian,* June 7, 1988, 34). The study found that at school computers were regarded as 'toys for the boys' and

many girls quickly became bored with them. Similar results have been found in Australia (Lamoureux, 1986).

THE EMANCIPATORY PARADIGM

The emancipatory paradigm involves making use of the computer as a labour-saving device, thereby reducing time spent on mechanical tasks. Some examples are:

1. Mechanical (electronic) calculations.
2. Word processing.
3. Desktop publishing.
4. Information retrieval - database or spreadsheet creation and manipulation.
5. Communications.
6. Music creation, recording and playback.
7. An electronic teaching aid. This occurs when the computer is linked to a large Video Display Unit or to a video projector and used as one would use a blackboard or an overhead projector to present a lesson.

Emancipatory Software Strengths

The strengths of emancipatory software are that:

- The student is in control and can make use of the computer as a tool as the need arises.
- Most of the available emancipatory software will supply the educator with a relevant and accurate resource.
- It provides students with a means to sort and compare large amounts of data easily and quickly.

- Data base software promotes deductive thinking and the development of correlation and cross-referencing skills (O'Brien & Kerr, 1986).
- It encourages the development of a variety of thinking skills, such as forming hypotheses and then manipulating, testing and controlling data in the process of proving or disproving data.
- Outlining or 'idea processing' software can assist students to develop a story outline before writing begins (Dailhou, 1986b).
- Graphics software can be used to create pictures which can provide a stimulus for writing, or which can be used to illustrate or even animate a story. (Dailhou, 1986b).
- Word processing software seems to promote the possibility of significant attitudinal changes to writing. According to Dailhou:

> The benefits appear to be that writers
> 1. Feel more in control of their writing, taking greater account of their audience and style because they can order and edit ideas easily.
> 2. Develop a sense of ownership.
> 3. Become involved in the actual process of writing.
> 4. Can think more clearly because they can always see their clean last copy; this appears to aid editing as their thinking is not impeded by messy hand writing.
> 5. Experiment to test the effect of changing a word, phrase, or paragraph.
> 6. Draft, redraft and edit in contrast to superficial rewriting.
> 7. Sit longer composing.
> 8. Appear to be inspired by the quality of their published work.
> 9. Make corrections because they are simple to make at any stage.
> 10. Will return to previously completed writings in order to update and to make alterations.
> 11. Feel tremendous satisfaction from controlling the whole process including the final printed copy.
> (Dailhou, 1986b, 100)

- Word processing software can overcome motor coordination problems, which may interfere with creativity, for children who find it difficult to write (Hall & Brown, 1986).

- Through the use of emancipatory software students are able to develop concepts about the operation of computers in society in general, and their advantages and disadvantages.

Emancipatory Software Weaknesses

The weaknesses of emancipatory software are that:

- Some of this type of software can be very difficult to learn to use, since it was not designed primarily for educational use.
- It is not always easy to find the right type of software for a given task. Indeed, the software may not exist for a particular task.
- The student needs to be aware that only a limited and closed set of information has been provided in any database they examine.
- The student can become over-reliant on the tool and make mistakes when entering data. This leads to incorrect results. The student may not then have the knowledge or skills necessary to check the accuracy of the results and to correct the error.

THE CONJECTURAL PARADIGM

The aim of this program type is to allow the student to manipulate and test his or her own ideas and hypotheses. It is based on the concept that knowledge is created through the student's own experiences and emphasises the exploration and manipulation of information. This exploration is done with the computer under the control of the student to the extent that the chosen environment permits.

The simplest level of conjectural software is the calculator level where the computer is used as a 'number cruncher' to help the student work through complicated arithmetic. This relieves the student of much of the drudgery of such work and allows better concentration on the task involved. The most complex level

is where the computer is used as a tool for modelling real life situations. This may involve the use of programming skills utilising programs such as LOGO, BASIC, or PROLOG to create your own program (to 'teach' the computer how to do a task). Some examples are:

1. Geometric shapes and mathematical modelling.
2. Economic modelling.
3. Societal modelling from historical fact and census information.

Much emancipatory software also has the capability to be used as conjectural software by adding a manipulation dimension to its usage. For instance, data can be stored in a database and retrieved from it. However, when the data is retrieved and manipulated, forecasts may be made, models may be constructed, or trends may be predicted. Information contained in a spreadsheet program, for example, may be used to calculate the increase in housing interest rates over the last five years, and then used to forecast the rates for the next five years.

Conjectural Software Strengths

The strengths of conjectural software are that:

- The student potentially has far more control over the computer than in any of the previous paradigms.
- The student can use this control to set up experiences which provide knowledge through those experiences. For example, by using a drawing and measuring program, the student may discover various rules of geometry, such as that the four sides of a square are each of equal length. During the course of these experiences, problem-solving skills are developed.

- It can be used, with programs such as LOGO, to generate language and concept development, even at Kindergarten level (Forwood, 1986; Frost, 1986).
- Programs such as LOGO can enhance selected geometric concepts such as angles and length, can facilitate the understanding of concepts such as algebraic variables and elementary algebraic formalisation (Oakley, 1986a), and can enhance the development of numerical modelling skills (Gilding & Pearce, 1986). James found in her research that the goals listed below would be reasonable for the use of LOGO at the primary school level.

    * Development of problem solving skills.
    * Development and reinforcement of mathematical skills appropriate to the age level.
    * Creative problem solving.
    * Divergent thinking.
    * Group co-operation.
    * Development of metacognition.
    * Use of precision in language.
    * Growth in confidence and self-esteem.

    (James, 1986, 193)

Conjectural Software Weaknesses

The weaknesses of conjectural software are that:

- The student needs to be reasonably familiar with the program being used, or else be provided with assistance by someone who is, whether that is a teacher or a peer.
- The student can be restricted in the environment which is able to be created by the limitations of the software being used.

The strengths of CAL software, as described above, are so because they are able to provide learning opportunities for students which would otherwise be difficult or impractical to provide in any other way. Instructional software allows the computer to be used as in infinitely patient individualised tutor. Revelatory

software allows the student to explore models and simulations of (sometimes dangerous) real world environments. Emancipatory software is able to reduce the amount of non-essential work the student must do manually to achieve learning objectives. Conjectural software allows the student to formulate and test and analyse hypotheses in a manner that is not available by other means. The weaknesses of CAL software, as described above, are so because of the limitations of the current computer environment. These may revolve around the difficulty or unsuitability of the subject matter for such an environment, or they may be a result of the current limitations of both the software and the hardware. The hardware limitations are decreasing apace as cheaper and larger memory capacity becomes available. The software limitations are also beginning to decrease both as a result of the improvements in hardware, and as a result of the fact that more and more educational software is being designed by educators rather than by computer scientists.

III
THEORIES OF LEARNING AND SOFTWARE DESIGN

As there are many definitions of learning, the intent in this chapter is to concentrate on those theories which are most closely related to CAL. Learning is a complex phenomenon encompassing many different types which range from the simple reflex response to the high level thought involved in solving complex scientific problems. Some theories of learning concentrate on simple behaviour, while others concentrate on the mental processes involved in complex learning. As yet there does not seem to be a single theory which satisfactorily encompasses all learning.

B. F. SKINNER

Software which was earlier described as belonging to the instructional paradigm, and being drill and practice type software, is based largely on Skinner's theory of operant conditioning (Bower and Hilgard, 1981). This theory grew out of observations of the performance of animals in a device that he invented, since called the Skinner box. This comprised a small box with a lever on one side. Whenever an animal in the box pressed the lever, the behaviour was positively reinforced by presenting the animal with food. Skinner isolated several low-level behavioural phenomena which he used as a basis for concepts used in analysing more complex forms of behaviour and its modification. Skinner's emphasis was placed on the reinforcement of a response, so that to ensure that learning occurs, a correct response must be followed by a positive reinforcement. Reinforcement

cannot follow unless the correct response is given. This differs from Pavlovian or classical conditioning in that, in the case of the animals in the Skinner box, it is not the stimulus of the lever but the response of pressing it which is correlated with the reinforcement.

Skinner's model of learning might therefore be described as:

Stimulus → Response → Reinforcement.

Since the emphasis in Skinner's model is placed on reinforcement, this became a complex issue for him. In order to learn a response, Skinner proposed that positive reinforcement follow every time a correct response occurred. Once the response was learned, he proposed that different schedules of reinforcement be used to maintain the correct response. In the first of these, reinforcement would follow the correct response at fixed intervals (fixed interval schedule), such as every two minutes or every four minutes and so on, regardless of how many responses were made during that period. Similarly, fixed ratio schedules provide reinforcement at every n^{th} response, while variable ratio schedules provide reinforcement on a random basis, varying either by time or by the number of responses made.

A Model for Instructional Software (Drill and Practice)

In applying Skinner's views to instructional CAL software, it is useful to examine Skinner's statement that: "The whole process of becoming competent in any field must be divided into a very large number of very small steps, and reinforcement must be contingent upon the accomplishment of each step." (Skinner, 1968, 21) This is exactly what is done in current drill and practice and tutorial software, where the subject material is broken down into small tasks which are generally presented one at a time. When the task has been mastered a reward is given to reinforce the learning and the next task in the

sequence is tackled. On the topic of reinforcement, Skinner says that:

> ...the human organism is, if anything, more sensitive to precise contingencies than the other organisms we have studied. We have every reason to expect, therefore, that the most effective control of human learning will require instrumental aid. The simple fact is that, as a mere reinforcing mechanism, the teacher is out of date. (Skinner, 1968, 22)

A computer would seem to be the ideal instrument for providing reinforcement as required. It can easily be programmed to provide positive reinforcement for each correct response until mastery is achieved. It can also follow Skinner's recommendation that once mastery is achieved reinforcement be shifted to an intermittent form rather than be continuous:

> The student will be less dependent on consistent and immediate reinforcement if he is brought under the control of intermittent reinforcement. If the proportion of responses reinforced (on a fixed or variable ratio schedule) is steadily reduced, a stage may be reached at which behaviour is maintained indefinitely by an astonishingly small number of reinforcements. (Skinner, 1968, 159)

A most effective reinforcing pattern would, then, seem to move from a continuous to a fixed ratio and finally to a variable ratio schedule of reinforcement. (Chambers and Sprecher, 1983)

Skinner's description of how to develop a programmed learning sequence (Skinner, 1958; Skinner, 1968; Bower and Hilgard, 1981) is directly applicable to instructional software. For Skinner, the important issues were:

1. Obtain a clear, detailed objective specification of what it means to know the given subject matter. For Skinner this typically means generating a detailed list of stimulus-response connections, usually in the form of questions and answers.
2. Write a series of stimulus (question) and response (answer) frames that expose the student to the material in graded steps of increasing difficulty and that frequently re-test the same facts from different angles.

3. Require the learner to be active; that is, require that a response be composed for each frame in the program.
4. Provide immediate feedback for each response (answer).
5. Try to arrange the questions in such a manner that the correct response is very likely to occur and be reinforced so that errors are avoided and learning is not usually accompanied by frustrating or punishing failures.
6. Permit students to proceed at their own pace.
7. Provide plenty of backup reinforcement (praise, points, tokens) for diligent and effective work on the program.

Drill and practice software designers generally apply these issues to their software. Step one is a necessity if any software is to be designed at all, as it would be impossible to develop a software specification without carrying out this step. Similarly for step two, a set of frames or screens must be developed in order to create a software specification. What may not be the case, however, is that the material is graded in increasing difficulty. Re-testing the same facts from different angles is not generally done at all in drill and practice software. For step three, all drill and practice software requires that the learner be active and respond to each frame of the program. Not all drill and practice software, however, provides immediate feedback for each response, and of those that do, the feedback is not always appropriate. Most current drill and practice software does not effectively carry out step five, and many are the exact opposite to the extent that a student can be 'stuck' in a particular frame if the correct response is not provided. As suggested in step six, students are almost always allowed to proceed at their own pace in drill and practice software, and most programs provide some form of backup reinforcement for effective work done in the program, as suggested in step seven.

R. M. GAGNÉ

Software which was earlier described as belonging to the instructional paradigm, and being tutorial type software, is based largely upon the information processing models of cognitive learning theories (Chambers and Sprecher, 1983). These models are concerned with how individuals gain knowledge and how they use it to guide decision-making and to perform effective actions. Such models try to understand the brain and how it works. Their methodology is to view the computer as a model of the brain and to employ much of the terminology and many of the concepts of information processing. A cognitive learning theory is concerned with several factors:

1. The effect of inputs or stimuli from the environment on the organism's receptors (visual, auditory and tactile).
2. The storage of information in the short-term memory of the central processor (the brain).
3. The storage of information in the long-term memory of the central processor.
4. The processes involved in encoding and decoding information.
5. The retrieval of the stored information, its possible combination with other data and its ultimate effect in the behaviour of the organism (Bower and Hilgard, 1981; Chambers and Sprecher, 1983).

Cognitive learning theories are directly applicable to the design and development of tutorial software. Robert M. Gagné has been an active pioneer in the development of this approach. Cognitive theory recognises that learner behaviour relies on external feedback, or reinforcement, although it does not accord as much emphasis to reinforcement as does Skinner. Of prime importance to Gagné is the identification of the goals of the learning task. "The best way to design instruction is to work backwards from expected outcomes. ... The basic

reason for designing instruction is to make possible the attainment of a set of educational goals." (Gagné and Briggs, 1979, 45)

In order to achieve these goals, specific instructional objectives should be developed. This task can be simplified by assigning objectives to "...five major categories of human capabilities." (Gagné and Briggs, 1979, 48) These five categories, as described by Gagné, are: intellectual skill, cognitive strategy, verbal information, motor skills, and attitudes (Gagné and Briggs, 1979).

To these categories, Gagné attached a set of executive control processes which "...select and set in motion certain cognitive strategies relevant to learning and remembering." (Gagné and Briggs, 1979, 154) Any single act of learning is presumed by Gagné to require nine different kinds of internal learning processing, and each is triggered by a different external instructional event (Gagné and Briggs, 1979; Chambers and Sprecher, 1983). They are:

1. Gaining attention triggers alertness and preparedness of the neural centre for incoming stimulation.
   ("Kim wrote a sentence yesterday in which the word 'walked' was spelt incorrectly.")
2. Informing the learner of the lesson objective triggers expectancy for data to be stored in short-term memory.
   ("Today we are going to look at how to spell the word 'walked' correctly.")
3. Revision or stimulating recall of previously learned data triggers retrieval to short-term working memory.
   ("Let's remind ourselves of the correct spelling of the word 'walk'.")
4. Presenting the stimulus material to be learned triggers selective perception which transforms data for short-term memory storage.
   (Data presented in a variety of ways.)

5. Providing guidance on learning the data triggers semantic encoding which is the process which prepares the data for long-term memory storage.

   ("Notice that 'walked' is made up of two parts - 'walk' and 'ed'.")

6. Eliciting the performance triggers the search and retrieval of data to the working memory and the provision of a response.

   ("Who can now spell the word 'walked'?")

7. Providing feedback about performance triggers the process of reinforcing correct learning.

   (Provide feedback.)

8. Assessing the performance triggers retrieval from long-term memory.

   (Test knowledge.)

9. Generalising triggers the transfer of learning and enhances long-term memory retention.

   ("Who thinks they can now spell the word 'talked' correctly if you are told that it follows the same rule as 'walked'?")

A Model for Instructional Software (Tutorial)

In applying Gagné's views to instructional CAL software, a model for the design of tutorial software could be illustrated as follows:

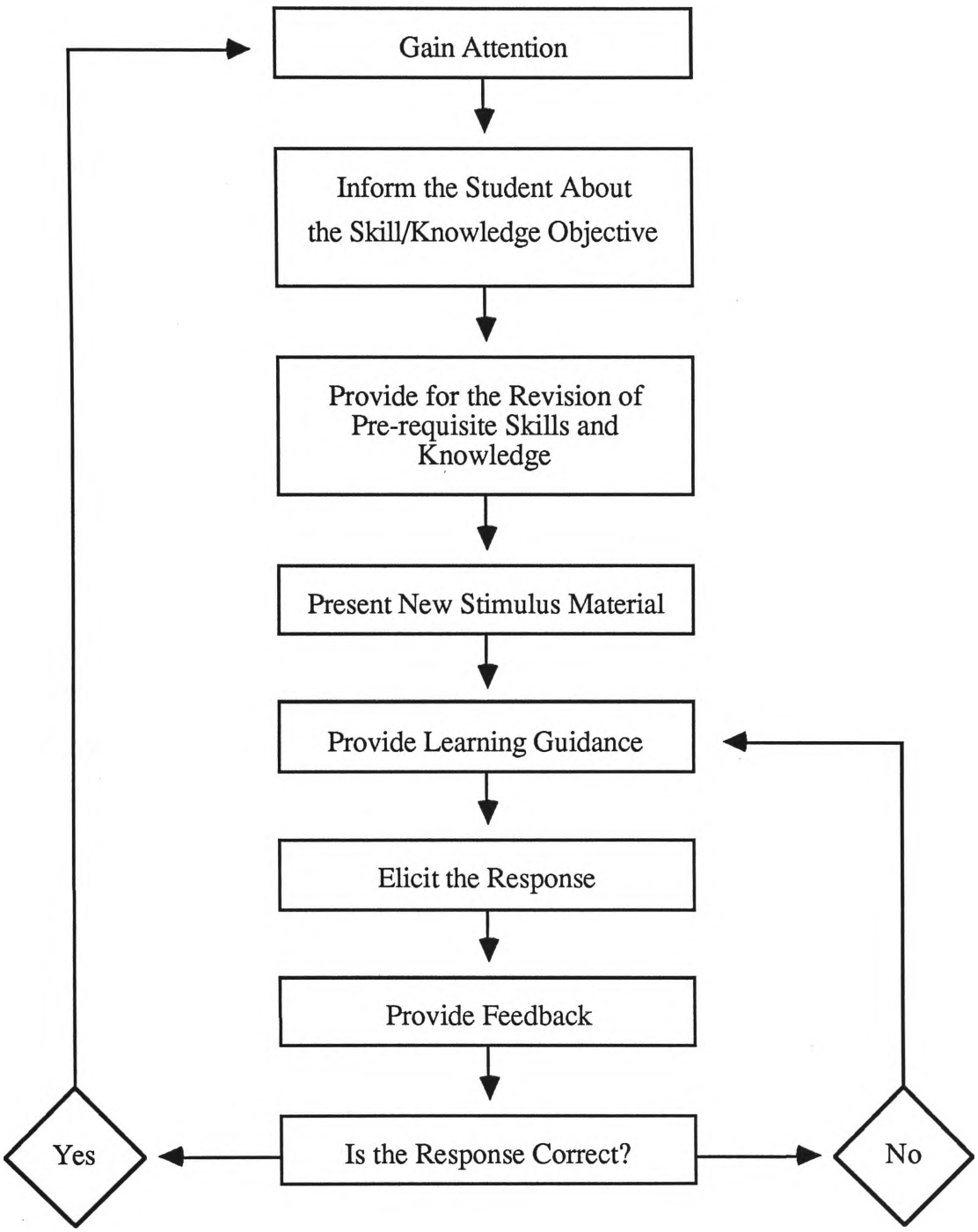


Figure 3.1 Model for Instructional software

Tutorial software designers generally apply these issues to their software. Although step one, gaining attention, may not often be carried out in the manner imagined by Gagné, it is often achieved intrinsically merely by the fact that the student is using a computer program to undertake some learning. Similarly for step two, most tutorial programs do not usually directly state the lesson objectives on screen for the student. They do, however, usually include these objectives as part of the supplementary literature provided with the program. While these objectives may be written for the teacher's information, the teacher is then able to make use of this information as seen fit, including informing students of the lesson objectives.

As for step three, most stand-alone tutorial packages do not provide for the revision of pre-requisite skills and knowledge within the program itself. This is the case because such revision may need to cover a vast range of skills and knowledge going back over several terms or even years. Also, the program designer has no indication of the readiness of any particular student to attempt a given tutorial program. That decision must be made by the educator involved. A description of assumed knowledge and skills is usually provided as part of the program package. There are many tutorial programs which form part of a series, such as the four programs *Addition, Subtraction, Long Multiplication*, and *Long Division Made Easy* (Greig & Stace, 1985). These tutorials do provide for revision of subject matter as part of the program. If, for instance, a student was using the *Long Multiplication Made Easy* program and was having difficulty with the addition section of the problem, then that student would be advised by the program to go to the *Addition Made Easy* program and revise the particular addition algorithm that was causing the difficulty.

Steps four through eight of the model are effectively carried out by the majority of tutorial programs. The steps with the greatest variability are step five, providing learning guidance, and step seven, providing feedback. Some tutorial

programs go to great depths to analyse the possible error the student may be making (Refer to Appendix B, for example) and to formulate a response that helps the student to realise what the error is and to overcome the error. Others simply provide a 'This is how to do it' mode which allows the student to see the solution effected but does not necessarily help understanding of the problem. As far as providing other types of feedback are concerned, there is again a wide range of types used, from the simple 'Right' or 'Wrong' type through to variable ratio rewards involving animation and music combined with error analysis and assistance as described above. As most current tutorial software really only cycles through steps four to eight, upon receipt of a correct response, the student is returned to step four rather than to step one of the model. An incorrect response is treated as in the model.

A. BANDURA

Software which was earlier described as belonging to the revelatory paradigm, and being mainly real-life modelling software, is based largely upon social learning theory, such as that of Albert Bandura. This theory is concerned with how personality could evolve out of social conditions. It deals with techniques of behaviour modification and personality assessment in both clinical and educational settings (Chambers and Sprecher, 1983). While the theory accepts the behaviouralists' concepts as being valid conditions for some types of learning, it has also proposed that a large amount of human learning is done through observing another person making responses and then by trying to imitate the responses of the model. "In the social learning system, new patterns of behaviour can be acquired through direct experience or by observing the behaviour of others." (Bandura, 1971, 3) Bandura further states his views as follows:

> ...most human behavior is learned observationally through modeling: from observing others one forms an idea of how new behaviors are performed, and on later occasions this coded information serves as a guide for action. Because people can learn from example what to do, at least in approximate form, before performing any behavior, they are spared needless errors. (Bandura, 1977, 22)

Bandura declares that observational learning is comprised of four component processes:

– attentional processes;

– retention processes;

– motor reproduction processes; and

– motivational processes.

Each of these is examined in turn below.

Attentional Processes

In order to be able to learn, people must be able to attend to, and perceive accurately the significant features of the modelled behaviour. These processes determine what is observed and what is extracted from the profusion of modelling influences to which one is exposed. The people with whom one regularly associates delimit the types of behaviour that will be repeatedly observed and learned most readily. Models who possess desired (although not necessarily good) qualities are usually sought, while those lacking such characteristics are usually ignored or even rejected.

In modern times, it can clearly be seen that the advent of almost universal access to television has greatly broadened the number and scope of models available. Today's television viewers can "... observe and learn diverse styles of conduct within the comfort of their homes through the abundant symbolic modeling provided by the mass media." (Bandura, 1977, 25)

Retention Processes

People need to remember observed behaviour in order to be influenced by it. Modelled behaviour thus needs to be represented in memory in symbolic form so that it can be recalled later. Bandura states that observational learning relies mainly upon two representational systems which he labels imaginal and verbal. An imaginal system is one whereby repeated exposure to modelled behaviour produces enduring, retrievable images. On later occasions these images can be summoned up. The imaginal system is mainly in use during early periods of development when verbal skills are absent or lacking, and when learning behavioural patterns do not readily apply themselves to verbal descriptions.

A verbal system is one whereby behaviour is encoded verbally, retained and is able to be reproduced at a later time. Bandura declares that most of the cognitive processes which regulate behaviour are verbal rather than visual. It is easier to remember the details of a route taken by use of a verbal code such as turn left, right, left, and right again than by reliance upon visual imagery. Such symbolic verbal codes facilitate learning and retention because "... they carry a great deal of information in an easily stored form." (Bandura, 1977, 26) Mental rehearsal of modelled behaviour also increases proficiency and retention.

Motor Reproduction Processes

The third component of observational learning concerns turning learned symbolic representations into actions. This is achieved by combining a spatially and temporally matched set of responses based on the modelled patterns. Many complex skills (swimming or driving a vehicle, for example) are best learned by being broken down into smaller sub-skills. This involves the refinement of behaviour through self-corrective adjustments on the basis of feedback, and from focussed demonstrations of sub-skills that have not yet been fully mastered.

Motivational Processes

Finally, motivation affects observational learning in that people are more likely to adopt modelled behaviours which result in outcomes they value, rather than those which have unrewarding or even punishing effects. High-status true-to-life models are more often imitated than low-status models which are divorced from reality. Bandura (1977, 50) sums up by stating that "... modeling influences can serve as instructors, inhibitors, disinhibitors, facilitators, stimulus enhancers, and emotion arousers."

A Model for Revelatory Software (Simulations)

The implications of Bandura's social learning theory are most applicable to simulation software (Chambers and Sprecher,1983). Despite the fact that real models are not used in such simulations, the computer provides an environment through which the student may learn vicariously through interaction with that environment. Reinforcement is provided by the changes invoked within the environment by student interaction with it. The student fully controls the situation and is thus positively reinforced. As Haynes states:

> The speed of such [computer-based] manipulation creates instant individual feedback, itself an invaluable learning tool, and, if Piagetian theory of construction of reality is correct, by increasing the amounts of trials one can make in a limited time one can increase the chance of generalization about effect, and thus of theory-construction. (Haynes, 1986, 52)

When creating such software, the important issues are:

1. Provide students with clear information regarding the content, structure and goals of the simulation and inform students of the benefits to be gained from the simulation. This will result, as Bandura would wish, in the development of expectations which serve to reinforce learning.

2. Include as much interaction between students and the computer environment as possible, allowing it to be used as many times as possible. This will enhance performance through self-correction and enhance retention.
3. Provide a model which is as life-like as possible.
4. Provide a model which is relevant to the student and which has the potential to directly affect his or her performance.

Simulation software designers generally apply these issues to their software. Although not all fully explain the benefits to be gained from the simulation, the content, structure and goals of the simulation are usually explained as these are vital to the successful operation of the simulation. Steps two through four are generally very well executed by current simulation software, and any limitations seen in these areas – particularly in providing as life-like a model as possible – are usually attributable to hardware limitations rather than software design limitations.

IV
THEORIES OF INTELLECTUAL DEVELOPMENT

In order to be able to construct software which is suitable for the students for which it is created, it is helpful to have a greater understanding of the cognitive processing characteristics of children. So as to develop such an understanding, this study will now examine other aspects of psychological research which can be translated into software design. The two main areas involved are developmental theories of intellectual performance and theories of cognition. In examining these areas, it may then be possible both to design better software and to perceive better the ways in which CAL may be used for the benefit of both the student and the teacher.

THEORIES OF INTELLECTUAL DEVELOPMENT: PIAGET

The object of Piaget's theory is to describe and explain the development of human knowledge. He is less concerned with the content of human knowledge than with its organisation and structure. Piaget sees development occurring in four stages (Piaget, 1969; Case, 1978: Parrill-Burnstein, 1981). The sensorimotor stage (birth to approximately 2 years of age) is where the child interacts with the environment at a motor level. The infant is born with grasping, sucking and visual tracking capabilities. As the infant acquires knowledge these capabilities are gradually applied to a wider and wider variety of objects and they become more and more coordinated.

The preoperational stage (2 to 7 years of age approximately) is where the child begins to represent basic forms of sensorimotor operations symbolically, and to manipulate objects mentally as well as physically. These representations can still only be applied in the presence of concrete objects. The concrete operational stage (8 to 12 years approximately) is where reasoning increases and certain acts are achieved that require more flexible thought processes. Children begin to acquire an understanding of the world that is more abstract and complex. Events are anticipated and consequences predicted with increasing accuracy. The formal operations stage (13 years to adulthood) is where abstract logic predominates. The child receives information regarding feedback, which is based only in part on the actual situation. Prior experiences are recalled through imagery and problems are solved through logical verbal and non-verbal reasoning (Parrill-Burnstein, 1981).

THEORIES OF INTELLECTUAL DEVELOPMENT: CASE

Case (1978) discusses a neo-Piagetian theory that bridges the gap between Piaget's theory of learning and an educational or instructional theory. Case suggests that instruction be designed to take into account aspects of the three components: structural analysis, assessment of the child's current operative functioning, and instructional design. Structural analysis involves:

1. Identifying the goal of the task to be performed.
2. Mapping out a series of steps by which a successful child might reach this goal.
3. Comparing this hypothesised series of steps with the performance subjects eventually exhibit.
4. Modifying the series of steps hypothesised and recycling through steps 1 to 3 above as necessary.
5. Testing the description for completeness.

Assessment of the child's current level of functioning involves:

1. Determining a question for which the child's answer would have been correct and then cycling through steps 2 to 5 above as though this were really the goal of the task.
2. Alternatively, assume that the child does not have access to some piece of information that appears to have been ignored. Hypothesise a rational procedure for reaching the goal and compare this hypothesised procedure with that which is actually used.

Instructional design involves:

1. Setting up a paradigm where the child may assess the effectiveness of the strategy which is currently being employed.
2. Demonstrating its inefficiency.
3. Assisting the child to discover why it is inefficient.
4. Facilitating the child's construction of a more adequate strategy.
5. Providing for consolidation and extension of this strategy.

THEORIES OF INTELLECTUAL DEVELOPMENT: BRUNER

Bruner (1971, 1974) considers his theory one of development and instruction. The child develops from an enactive representation of the world, requiring total motor involvement, through an iconic representation, where the child begins to use imagery to take the physical positions of others, to a symbolic representation enabling the mental manipulation of abstract symbols. The child develops and applies constructs to the social world. Bruner believes that schools should be active facilitators of learning, and that rules rather than specific skills should be learned. He further believes that teaching should be directed to the child's present developmental level and should convey new rules in progression from simple to more complex.

THEORIES OF INTELLECTUAL DEVELOPMENT: GAGNÉ

While Gagné's cognitive learning theory has already been examined in chapter three, Gagné and Briggs (1979) further state that there are also conditions which are necessary for learning to occur. They are:

1. Discriminations. The child must be able to differentiate between stimuli on the basis of physical dimensions.
2. Concrete concepts. The child must be able to classify a group stimulus on the basis of common characteristics.
3. Defined concepts. The child must be able to attach and demonstrate the meaning of some particular object, event, or relationship to the concrete concepts.
4. Rules. The rules must be learned and the child must be able to show them to be learned by responding consistently to regularities occurring across a variety of situations.
5. Higher order rules, which are complex combinations of simple rules, must also be learned.

INFORMATION PROCESSING

Information processing is a theory of learning which has recently been proposed. Cognitive information processing is the method by which information about the world is analysed and synthesised in sequential steps (Neisser, 1976). The cognitive processes involved are: attention, concept organisation, memory, language, cognitive mapping, and social cognition (Parrill-Burnstein, 1981). Wright and Vliestra (1975) discuss the development of systematic information processing in children. The information processes of exploration and search, which are reflected in observing behaviours, are the means by which the organism acquires information from the environment. How this information is acquired is what

distinguishes these two processes. Exploration is motivated by curiosity and guided by stimulus salience. Exploration:

1. Is more spontaneous and less systematic;
2. Consists of shorter sequences;
3. Shows less continuity from one sequence to another;
4. Is more divergent; and
5. Is influenced by salience of stimulation.

Search, on the other hand:

1. Is more task- and goal-oriented;
2. Consists of instrumental responding; and
3. Is of two types — perception and logic.

Wright and Vliestra propose that the development of both search and exploration progresses from general processing to more specific and systematic processing.

ATTENTION AND MEMORY

Wright and Vliestra (1975) have found in their studies that pre-school children tend to attend to the most salient characteristics of a stimulus, to position cues, and to random items. Children of 5 to 7 years scan a visual array more systematically, although the scanning is still erratic. Children of 10 to 14 years increase instrumental or instructional learning and recall more central or task-relevant information. The ability to maintain central task performance by excluding certain incidental information improves with increasing age. However, "... if [attention] strategies are interfered with by the imposition of a distraction task central performance of older children is reduced to a level of performance similar to that of children several years younger." (Hagen and Kail, 1973, 170)

Broadbent (1958) assumes that irrelevant information is filtered out at multiple levels of processing. He suggests that the perceiver is limited in the capacity to

process all peripherally available information. Furthermore, what is processed is processed through a single processing centre. Neisser (1976), on the other hand, criticises the assumption of a single processing centre and a limited capacity to process information. He notes the ability to learn new information throughout life, as well as the ability to process simultaneously or attend selectively to information for more than one modality or within the same modality. Neisser also proposes that unwanted information is not filtered out — it is simply not attended to. The internal organisation of information and the available stimulation determine what information is perceived.

Since the computer is easily able to employ graphic representations (and a great many programs do), it is interesting to note some of the research done in this area. Farnham-Diggory used logographs such as those in Figure 4·1 in an effort to bypass perceptual confusion in young readers resulting from their efforts at alphabetic discriminations. This work found that 3 year olds could learn the symbols easily and could easily read sentences composed of symbols, such as *jump over block,* or *jump around teacher.* When asked to demonstrate the meaning of the sentence, however, problems arose. Children up to 5 years old demonstrated each item individually. They *jumped* on the floor, they made a sign in the air representing *over* and they pointed to a block on the floor. They failed to integrate or synthesise cognitively the separate chunks of information (Farnham-Diggory, 1972).

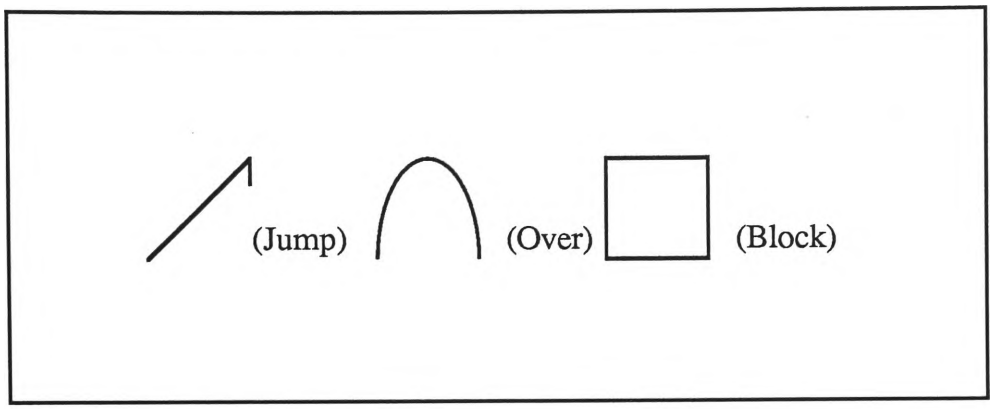


Figure 4.1 Reading logograph

In order to be able to demonstrate the meaning of the sentence, the child must be able to hold in mind simultaneously the three symbolic concepts and some notion about real-world events which match those concepts, such as *jumping* and *blocks*. Unless the child can mentally survey these five or six information chunks simultaneously, the child will not detect the correct relationships among them. This task requirement is beyond the apprehension capacities of a typical 5 year old. Apprehension span at that age is a solid four chunks and at 6 to 7 years of age apprehension span is five chunks (Farnham-Diggory, 1972).

Myers and Perlmutter (1979) found in their studies that children as young as $2^1/_2$ years recognised approximately 80 per cent of to-be-remembered items as seen previously. Recognition was excellent for children as young as $4^1/_2$ years. As well, 4 and 5 year old children recognised new items and previously shown (old) items equally well, while the youngest children had greater difficulty recognising items as new or old. Myers and Perlmutter suggest that the age difference in recognition observed between children between the ages of 2 and 5 years was the effect of retention rather than changes in decision strategies. Cues such as part/whole relationships, habitat, item associations, or spatial layouts benefitted the older children more than the younger children. Moely found that when sorting items for recall, "...the younger child tends to divide a set into more categories, with

a smaller number of items in each category, than the older child uses." (Moely, 1977, 215)

Hagen (Farnham-Diggory, 1972) conducted short term memory task tests on four groups of children of different age groups: a 6 - 7 year old group; an 8 - 9 year old group; a 10 - 11 year old group; and a 12 - 13 year old group. Hagen used Broadbent's model (Broadbent, 1958) of filtering mechanisms and found that central memory task performance increased as a function of age, whereas incidental performance did not and actually declined at the oldest level. Hagen further found that information overload results in a decrement in performance of central memory to the same extent for the oldest as for the youngest children. Among the youngest children, those who did well on central memory also did well on incidental memory. At the oldest level, however, those who did well on central seemed to ignore the incidental information. This finding supports the hypothesis that the ability to maintain task performance by excluding incidental information improves with age.

Hagen also tested children aged 6, 7, 8 and 10 in an attempt to discover whether verbal labels mediated memory (Hagen, Jongeward & Kail, 1975). He found that there was an increase in memory performance as a function of increasing chronological age, and that overt verbal naming resulted in increased memory performance for the three youngest age levels, but not for the oldest. When 5 year olds were trained to rehearse with labels their memory performance was facilitated. Children at age 10 and older spontaneously employ rehearsal in order to remember, but when they were tested and asked to consciously label, a decrement in overall central performance resulted (Hagen and Stanovich, 1977). However, "...older children are better able to employ cognitive skills in order to concentrate exclusively on task-relevant information, as compared with younger children." (Hagen and Stanovich, 1977, 98)

Flavell and Wellman (1977) discuss the development of metamemory, which is described by Flavell as an individual's knowledge or awareness of memory processes, such as storage and retrieval. Flavell found that 11 year old children, when told to remember information, were aware that implicit in these instructions was to do something to the material presented so as to remember it better. They spontaneously rehearsed or categorised the items during presentation. Seven year old children, on the other hand, recognised that something needed to be done but did not employ any techniques to improve memory. Four year old children neither realised that something needed to be done nor instituted any procedure to facilitate memory (Appel et al., 1972). Flavell and his associates (Flavell, Friedrichs, & Hoyt, 1970) found that 6 year old children could be trained to rehearse to-be-remembered material, although they did not spontaneously do so. Although this induced rehearsal facilitated children's recall performance, they only rehearsed when told to do so and performance returned to baseline when the children ceased rehearsing.

As early as age 2 years, items are clustered semantically and the recall of one item can cue the recall of another (Flavell and Wellman, 1977). Between the ages of 3 and 5 years, items are organised on the basis of taxinomic clusters (for example, apples and oranges are both fruit). Older children code information as it relates to serial order. Paris and Lindauer (1977) found that older children remember stories better and can correctly answer all categories of questions more often that younger children. They further found that "...[better] retrieval from memory appears to improve with age." (Paris & Lindauer, 1977, 44) Cole, Frankel, and Sharp (1971) studied the development of free recall in children and found that recall improved when the items presented were visual, written, and grouped according to semantic categories. Kail and Siegel (1977) found that the number of items capable of being stored in iconic or visual memory was comparable for 5, 8, and 11 year olds and

adults, and that the rate of decay from iconic memory was very similar for 5 year olds, older children and adults. The final words on memory are left to Bruner who states that discovery learning enhances memory: "...the very attitudes that characterize figuring out or discovering things for oneself also seem to have the effect of making material more readily accessible in memory." (Bruner, 1974b, 412)

CONCEPT ORGANISATION

Gholson, Levine and Phillips (1972) state that pre-school and kindergarten children tend to concentrate on a specific aspect or position of a stimulus during many concept organisation tasks. They do not solve the problems presented, although their response choices are systematic and orderly. Between the ages of 5 and 7 years, a transitional period occurs during which inconsistent or random responding increases (Rieber, 1969), physical quantities are not successfully manipulated and literal interpretations of stimuli are made (Piaget and Inhelder, 1969). Between the ages of 7 and 8 years, the child begins to respond to specific stimulus aspects and formulates plans that lead to solutions to problems (Bruner, 1974; Gholson and McConville, 1974). The responses of the child become more logical, and the consequences of these responses become more predictable and anticipated (Piaget and Inhelder, 1969). After 8 years of age, selective attention to dimensions occurs, and the frequency of the child's solutions to problems increase, as does the efficiency of the strategy employed (Gholson, Levine and Phillips,1972).

At about 10 years of age, children use more sophisticated strategies and, on certain tasks, perform similarly to adults (Parrill-Burnstein,1981). Also, at about this time, attention to central information increases (Hallahan, Kauffman and Ball, 1973) while attention to incidental information decreases (Hagen, 1967). Accurate cognitive maps (mental images) are formed (Hardwick, McIntyre and Pick, 1976),

mutual role-taking occurs (Shantz, 1975), and responses to feedback are more appropriate as children learn to retain a response when correct and change a response when told it is incorrect (Gholson, Levine and Phillips,1972). The efficiency of the strategies used increases with age, as older children solve more problems in fewer trials.

Bruner (1974c) suggests that the child uses strategies to solve concept learning problems. Initially the child identifies temporal and spatial similarities in the environment. More efficient problem solvers learn to focus on more relevant and abstract relationships while ignoring the irregularities or inconsistencies that occur in information. This increase in efficiency is a function of both the problem solver's underlying plan and cognitive processing capability. Bruner further states: "Practice in discovering for oneself teaches one to acquire information in a way that makes that information more readily viable in problem solving." (Bruner, 1974b, 406)

LANGUAGE

Early infantile language consists mainly of crying, cooing and babbling (Hurlock, 1964). By the age of twelve months children usually begin to label objects, and by eighteen months most children can combine two words. Carey (1978) estimated that between the ages of eighteen months and 6 years, children learn approximately nine new words a day, which means that by the age of 6 years the child has learned over 14,000 words. Generally nouns occur with the most frequency (Dale, 1976). Between the ages of 2 and $2^1/_2$ years, extraction and generalisation of syntactic rules begins to occur to a limited extent. By approximately 3 years the child's lexicon is represented as is the adult's (Carey, 1978) and the child can produce simple sentences of up to ten or eleven words in length. By $3^1/_2$ to 4 years children can use adverbs, adjectives and other auxiliaries

and begin to progress rapidly in the development of sentence structure. Between the ages of 4 and $4^1/_2$ years, grammar is similar to adult speech. However, the average 5 year old does not as yet have the concept of a word, which has clear implications for learning — most notably, learning to read. "If one does not know what a word is, obviously it will be difficult to learn what the convention of spaces in text represents." (Carey, 1978, 53.) By the end of the sixth year, the child's language is essentially complete and accurate in both structure and content. Events can be verbalised sequentially, conclusions can be drawn and inferences can be made. Miller (1977) estimated that between the ages of 6 and 8 years, children learn over twenty new words a day. Between the ages of 7 and 8 years ideas begin to be shared verbally and by 12 years of age, parallel changes in language content and cognitive development occur (Parrill-Burnstein, 1981). Sex differences in information processing performance appear after puberty. Females tend to obtain higher scores than males, on average, on certain verbally-loaded tests and males tend to obtain higher scores than females, on average, on certain spatially- and/or numerically-loaded (intelligence) tests (Vernon, 1987).

COGNITIVE MAPPING

Cognitive maps or mental pictures of our environment underlie much of our problem solving ability (Boulding, 1961). Cognitive mapping skills do not develop in isolation, but are a part of the child's natural cognitive development (Parrill-Burnstein, 1981). Kaplan (1973) suggests that cognitive mapping skills enable decisions to be made that allow us to adapt. He describes four types of knowledge relevant to cognitive mapping:

1. Where one is;
2. What is likely to happen next;
3. Whether what happens will be good or bad; and

4. What are the possible courses of action.

Cognitive maps affect the abilities to move in space, to form mental images, to take another's perspective, and to infer the roles of others. Cognitive mapping skills assume adequate motor movement, selective attention, and memory.

SOCIAL COGNITION

Social cognition begins with the processing of information relating to feelings, beliefs, and attitudes (Flavell, 1977). The manner in which these items are accumulated, related, and integrated is called social cognition. A child's social cognition is reflected in observable behaviour, which can be called the product of social cognition. The three products of social cognition are symbolisation, conceptualisation, and social competence (Parrill-Burnstein, 1981). Social competence is the ability to interact appropriately with others and to interpret the behaviour of others accurately. In order to be socially competent, the three skills of role taking, person perception, and perspective taking are basic. Role taking is the ability to understand the social position of others (Flavell, 1977), person perception is the ability to identify and process relevant cues about others (Shantz, 1975), and perspective taking is the ability to view objects from another's perspective (Hardwick, McIntyre & Pick, 1976). The development of social cognition parallels that of cognition, which has been discussed earlier in chapters three and four.

ADDITIONAL INSTRUCTIONAL CONSIDERATIONS

According to Winne and Marx (1982) there are three basic cognitive processes that learners use in instructional settings: comparing, generating, and re-examining. These three cognitive processes are probably inseparable from each other and to achieve a cognitive product such as comprehension, all three are probably applied in

varying amounts and sequences. Comparing aspects of content to promote learning entails:

1. Unspecified comparing of lesson content. This comparison is not guided by the teacher.
2. Comparing codes for a single concept. Look at different ways of coding or representing the same information (e.g.: $^1/_4 = 0.25$).
3. Comparing attributes across two or more concepts. The focus is at the microscopic level (e.g.: the individual words in a grammar lesson).
4. Using rules. Apply known rules to new examples.

Generating a new way to represent content and to generate new content per se based on the information provided entails:

1. Generating codes as an aid to remembering information (e.g.: mnemonics).
2. Creating hypotheses, testing them, and trying to generate new rules to be applied.

Re-examination entails using metacognition, that is, thinking about the processes and outcomes of thinking. The processes involved are:

1. Monitoring the processing of content.
2. Appraising the information learned to monitor the extent to which content has been mastered.

Peterson and Swing (1982) identified many cognitive strategies employed during classroom instruction. These included repeating and reviewing information to oneself, relating information to prior knowledge, anticipating an answer to a teacher's question, trying to understand the teacher or figure out a problem, checking one's answer with the teacher or a student, reworking a problem in one's head or on paper if the answer was incorrect, reading or re-reading directions or

problems, and motivating oneself with self-thoughts. The employment of such strategies would indicate active processing of the information to be learned. Indeed, Peterson and Swing state that "... such active manipulation must occur for the information to be learned." (Peterson & Swing, 1982, 487)

V
SOFTWARE EVALUATION

The need for a set of guidelines for evaluating educational software is emphasised by the general lack of computer expertise and software experience among the teaching community. This can result in the purchase and use of software which is of questionable quality and which may be unsuitable to the curriculum being used by that system. An example of the difficulties involved is the wide variation to be found between countries and, indeed, states in the language and methods used in the simple Mathematics operations of addition, subtraction, multiplication and division. Much current software suffers from one of two (or sometimes both) problems. The software may be designed and written by programmers who have little or no educational expertise. Such software may lack sound educational aims and may make use of poor teaching techniques. Other software may be designed and written by educators who have little or no programming expertise, so that the software is technically flawed. There needs to be available a system such that the classroom teacher is able to conduct evaluations so that worthwhile software is purchased and used.

The current system in most states is that software is evaluated by selected computer literate teachers (such as those selected by the Computer Education Unit in New South Wales, or by Schools Computing Services in Queensland) who then prepare summary evaluation forms which are made available to classroom teachers. The main problem with this approach is that only a brief summary of the evaluation is included in the released software evaluations. If the classroom teacher conducts an evaluation, he or she can not only look at the merits or demerits of the software

itself, but can also decide whether or not it is applicable to aspects of the curriculum being taught by that teacher. This can be a highly personal thing. One teacher may find a particular program highly useful, despite recognised flaws, while the teacher in the next classroom may prefer a different program to treat a similar topic.

In order to conduct software evaluations, it is preferable to involve several people from different locations (schools) and backgrounds who should conduct both personal evaluations and field-test the software. It is also preferable that the evaluating personnel include teachers who are knowledgeable in the particular curriculum area involved and teachers who have some computer expertise and software experience. When evaluations have been completed, the results should then be synthesised and compiled to give a well-balanced evaluation profile of the software.

When evaluating software, the use of a form (such as that shown in Appendix A) will facilitate and unify the evaluation process. The aspects to be assessed on such a form could include items such those discussed below.

GENERAL PRODUCT DESCRIPTION

This section of an evaluation provides a general overview of the particular piece of software being evaluated. The information gathered concerns items such as the program's name, publisher, and whether it is a stand-alone program or is part of a series of programs. There is a need to know on which computers the program will run, its price and the publisher's backup policy. This information is important should revised and improved versions of the program be released in the future. It is also useful to have some information about the reviewer, so that a reader may feel more or less confidant about the results obtained by the reviewer, depending upon the reviewer's qualifications to conduct such a review.

PROGRAM TECHNICAL REQUIREMENTS

This section provides a more specific view of the particular piece of software being evaluated from a materials and hardware point of view. The information gathered concerns such items as the contents of the software bundle. These could range from a package which contains a disk only right through to a package which is a complete teaching kit containing disks, operating instructions, teacher guides, teaching aids such as stencils, posters, stickers and so on, as well as suggested lesson plans. Other information gathered here concerns the hardware requirements needed to run the program. Such things as whether or not a printer, a disk drive, or a colour monitor is a vital requirement for successful operation of the program and what the minimum memory required is to support the program need to be discovered.

PROGRAM DESCRIPTION

This section provides a specific description of the particular piece of software being evaluated in the areas of program type and program usage. The program type

criteria broadly identifies the pedagogical method used by the program, as discussed earlier in the section on CAL types. The target audience should be clearly identified by the publisher as should the curriculum area(s) and topic(s) covered by the program.

EVALUATION CHECKLIST

Content

This section of an evaluation provides a more detailed examination of the actual items considered in the evaluation of the product, in checklist form for ease of reading. These are grouped (on the form in Appendix A) under the same headings as those used in the Evaluation Summary. Specific educational objectives for the program should be fully and clearly stated. The objectives should describe the knowledge and/or skills the student is expected to have mastered upon successful completion of the program, or of a lesson within the program. The stated objectives should be educationally sound and should be relevant to the curriculum area being treated. The objectives should be achievable by the target audience which should be clearly defined whether by age, class level, skill level, or other means.

It is particularly important that the vocabulary level used within the program is appropriate to the defined audience. It is important that this is checked, not only in the educational or instructional aspects of the program, but also in the operational aspects. Necessary instructions for using the program need to be clearly understood by the user as do any reward or feedback messages on student progress.

The content of the program should also be checked for its accuracy, whether it is appropriate for the target audience, and whether the instructional method chosen is appropriate for the content and for the chosen audience. The content should also be "...free of sex role stereotyping, racial or social bias [and should facilitate]

progress to more equal educational outcomes." (Teaching Learning And Computers, 1983, 43). The New South Wales Department of Education has prepared an Identification Checklist which can be used to help identify bias in software. It is reproduced below. Not all of the questions posed are applicable to every piece of software. Nor should software that doesn't meet bias-free criteria be rejected out of hand. Appropriate steps can be taken by the teacher to overcome the bias identified by the evaluation process.

WHEN ANALYSING SOFTWARE ASK YOURSELF:

* Does the program present positive images and role models for students?
* Is the diversity within groups reflected?
* Does the program present a non-violent point of view?
* Is the language appropriate and non-disparaging?
* Which groups of people are omitted or ignored?
* Which people are shown as having stereotyped characteristics and roles?
* Which groups of people are seen as being active and or passive?
* Which groups of people are seen as being dominant/subservient?
* Which people are seen as being superior/inferior?
* What values and attitudes are being rewarded or reinforced?
* Is the inclusion of groups representative or trivial?
* Are the role models portrayed trivialised or valued and respected?
* What kind of language and/or images are used in the advertising/documentation as part of the software package?
* What is its purpose? What are its possible effects?
(Handle With Care, 1985, 3)

Program Features

The program needs to be robust in the sense that it is expected to perform reliably, probably on a daily basis, without experiencing any malfunctions. It should also be designed so that it is thoroughly foolproof against unexpected keyboard activity, whether that be deliberate or accidental. At the same time, the program should be easy for the intended audience to use.

Since most educational programs released to date rely heavily on the visual display alone, all text and graphics should be clear and easily read and understood by the intended audience. Particular attention needs to be paid to whether or not a

program designed to run primarily on a colour monitor will run effectively on a monochrome monitor without appreciable loss of visual clarity.

The type of feedback given by a program can vary widely, from a simple "Right" or "Wrong" to the over-use of extravagant graphics and sound displays. Whatever feedback is used, it should be appropriate both to the program and to the intended audience. Instructional programs such as drill and practice programs are likely to employ a simple level of feedback, while more complex conjectural programs such as simulations may rely for their effectiveness on the student fully understanding and reacting correctly to the feedback supplied. Feedback should not be used to the extent that it becomes more important than the instruction and thus is effectively a distraction. Nor should feedback be unnecessarily supplied as this may become tedious and retard the student's progress through the program or lesson.

The student should be in control of the rate of progress wherever that is possible. Students should be cueing the program to continue when they are ready. They should not be moved on at some arbitrary time decided by the program. Most programs achieve student control by allowing the student to press a special key such as the **spacebar** or the **return** key to signify that the student is ready to go on to the next item. Some programs are designed to be used with the teacher playing an active role in the program's presentation, but many are designed to be used independently by the student. This is a most important feature in the typical situation where there are only a few computers available – or only one – at any given time so that only a few students have access to them, while the remainder work on other things. Where more computers are available as in a laboratory situation, for example, an enriched educational environment can be developed. By "allowing the drudgery of rote learning and the answering of discovery-mode questions to be handled by computer assistance, teachers are freed to give more time to become better teachers." (Sale, 1982, 69)

Special Features

Although not applicable to all programs, the availability of special features can enhance the ease with which a program can be used. Whenever a student is unsure as to how to proceed or how a certain function or feature works, many programs allow the student to push a key such as the **?** key or the **Esc** key to obtain on-screen assistance in working with the program. Some programs also provide the capability to print out lessons, tests and other items that have been created using the program. Indeed, some programs are developed solely for the purpose of producing art work such as certificates, cards, pages with printed borders, and so on. While these programs are not CAL programs but may be termed utility programs, they can be used by both the teacher and the student for utilitarian purposes such as those described.

Another useful feature available in some programs is a program editor which makes it possible to make changes to the original program to suit local needs and conditions. Typical examples are the way in which the date is displayed in some programs, either 6/30/88 (American standard) or 30/6/88 (Australian standard), and the spelling of words used within the program, either color (American standard) or colour (Australian standard). A feature such as this helps to make the program fit local curriculum requirements as far as the educator is concerned, and allows the program to cover more easily a wider market base as far as the publisher is concerned.

Classroom Application

Generally speaking, the less preparation required by the teacher or the student, the more convenient a program is to use. If a program is one of a series, however, there may be a pre-requisite that certain other programs or lessons have been successfully completed as part of the preparation for undertaking the current

program or lesson within a program. Some programs keep records of progress, where appropriate, while others notify of progress but do not record it. It may be up to the teacher to establish a record of programs or lessons completed and to advise on which program or lesson to attempt next. In a busy classroom where only one or two computers are available, convenience of use is an important factor in order to make optimum use of the technology.

Support Materials

As many teachers using the program will still be novices in the use of computers, it is vital that the User Guide provides sufficient support to the teacher to be able to use the program effectively. No learning experience occurs in isolation, so the program package should include teaching ideas, activities and support materials which help the teacher to integrate the program with the curriculum. It is important that these teaching ideas, activities and support materials also fit the curriculum being used.

EVALUATION SUMMARY

This section of an evaluation form (as in Appendix A) provides a brief summary of the actual evaluation that has been carried out, using a common numerical code such as 5, 4, 3, 2, 1. An overall impression rating can then be given at the end of this section. In the case of the form in Appendix A, it is the total gained from Part E of the form.

VI
SAMPLE EVALUATIONS

The following sample evaluations are broadly based on the sample evaluation form shown in Appendix A, and on the items discussed in Chapter V.

Addition Made Easy

Program Description

The first program to be evaluated is *Addition Made Easy* (Greig & Stace, 1985a). This program was designed for the Apple // series of computers, requiring a basic memory of 48K. The version number evaluated was 2·2. The program is provided in a plastic covered three-ring binder with sleeves inside both the front and back covers, which contain the two disks (one Teacher Disk and one Student Disk) as well as a laminated Quick Reference Card for the student, and reward stickers. In order to be able to operate this program, a single disk drive is essential. Optional hardware items which could be used include a printer for printing out sets of problems, a second disk drive which could be used for the creation of student files, and a colour monitor which would enhance the visual appeal of the program.

The program can be operated in any one of three modes. The first is called Stepthrough, and is really a blackboard-type demonstration of how a problem is done. A vertical addition problem of any kind can be displayed on the screen either by using a menu to generate the problem or by entering it manually. By pressing the **return** key, each step of the problem will be shown on the screen. By pressing the **H** key, the descriptive help available at each step can also be displayed as needed, or the teacher might describe the process verbally if the stepthrough mode

is being used to introduce a new addition concept. The second mode of operation is Tutorial mode, where two levels of help are provided at every step of the addition process. An addition problem is displayed on the screen and the student is expected to type in the correct part of the answer. The insertion point cursor always locates itself at the next correct position after each part of the answer has been entered. If the student is not quite sure what is expected, helpful hints are provided when the **H** key is pressed. If the student cannot correctly answer any step, the program does so automatically, explaining what needed to be done, and moves on to the next step in the problem. The third mode of operation is Test mode, where the program functions as it does in Tutorial mode, except that no help is provided. The aim of this mode is to provide a means of testing the knowledge and skills gained by the student in the solution of addition algorithms.

A bonus of the program is that it provides a diagnostic tool for both the teacher and the student. The program is suitable for use by individuals from the age of approximately 7 years, groups, or classes, and is also suitable for use as a remedial tool for older students and even adults. Once an addition concept or skill has been introduced by the teacher, this program is suitable for concept development and review, as well as skill development and review. "The ADDITION MADE EASY package is not designed to introduce the concept of addition. The introduction of this concept and the layout and steps required in the process must first be handled by the teacher." (Greig & Stace, 1985a, 2)

The program treats addition algorithms, in vertical form only, from basic addition problems such as:

```
7 +
6
```

through to complex additions such as:

```
6473 +
 780
7459
  38
```

Full descriptive information about what is needed to be done in order to complete each step of the algorithm is always available should it be needed, simply by pressing the **H** key.

Content Evaluation

The program documentation clearly and fully states the objectives of the program, as follows:

> This package addresses the problem of developing confidence in the application of the addition process. The aim is to show that even the most complex addition problem can be completed by **the step by step application of simple operations**. Once children have mastered this step approach they will always have a reliable "fall-back" method for lifelong use. (Greig & Stace, 1985a, 2)

This objective is both worthwhile and relevant to the curriculum. The target audience is clearly defined and the vocabulary used is at a level appropriate to that audience. "The level of concepts and language used in this package has been designed for children between the ages of seven and ten years." (Greig & Stace, 1985a, 2) The program's content is accurate, appropriate and free from bias. The teaching method used is appropriate. Using the rating scale as shown on the form in Appendix A, the content of *Addition Made Easy* is awarded four points.

Program Features Evaluation

The program is fully error trapped and 'bug' free. It cannot be accidently stopped or interrupted. The presentation of the information provided in the program is clear and easy to understand. The feedback used is appropriate and not a distraction, and is based on a randomised variable ratio reward schedule. The rate of progress through the lesson is established by the user, and the program can be used independently without input from the teacher if desired. Using the rating scale as shown on the form in Appendix A, the program features of *Addition Made Easy* are awarded five points.

Special Features Evaluation

The program provides a comprehensive Help facility which is available for every step of every problem when in Tutorial mode: "... help is available in exactly

the same natural manner as it is in the classroom, that is, before they [the students] try a difficult step, not after they have gone wrong when it is usually too late for help." (Greig & Stace, 1985a, 3) There are two levels of help provided. The first advises the student of what needs to be done (e.g., "You need to add the 4 to the 3 and type the answer"), the second advises the student of what needed to be done and then carries out that step and moves on so that the student cannot become stranded at any given step. This version of the program also includes a print out facility, which allows the creation of individually tailored sets of problems, which may be done in class or taken home for homework. Using the rating scale as shown on the form in Appendix A, the special features of *Addition Made Easy* are awarded five points.

Classroom Application Evaluation

Although the program can be used without any teacher preparation, by using the Teacher disk in class, the ideal method is for the Teacher to create sets of problems for individual students on a Student disk. "The system is based upon the establishment of files of exercises by the Teacher. These files may be created for individuals or for groups of students ... it is possible that a whole school term's work in addition practice could be prepared on disk." (Greig & Stace, 1985a, 3) The teacher can either do this manually by typing in the problem set, or the program on the Teacher disk can automatically generate sets of problems as required by each student. The students then takes the Student disk and do the problems, "... at their own pace, with as much help as **they** request." (Greig & Stace, 1985a, 3) There is no preparation required by the student. As the program can be used independently, it is very convenient to use in the classroom. The teacher could also use it as an electronic blackboard when introducing the concepts involved. The student's progress through the program is tracked by the program and the results are displayed at the end of the set of problems for the information of the student and, as

necessary, of the teacher. The results are not recorded on disk. Using the rating scale as shown on the form in Appendix A, the classroom application of *Addition Made Easy* is awarded five points.

Support Materials Evaluation

All User Guide instructions are adequate and correct, and the educational design of the software is accurate. There are no supplementary activities suggested in the packaging, nor are any teaching ideas provided. This is, perhaps, the only real shortcoming of the package. The packaging is sturdy and convenient to use and store, and should last a long time in the tough classroom environment. Using the rating scale as shown on the form in Appendix A, the support materials of *Addition Made Easy* are awarded three points.

Summary

This gives a total of twenty two out of a possible twenty five points to this software package, making it one that is an excellent package that could be recommended and used without hesitation.

Dragon World

Program Description

The second program to be evaluated is *Dragon World* (Matson, 1985). This program was designed for the Apple //, B.B.C., and 480Z series of computers, requiring a basic memory of 48K. The version evaluated was the Apple version. The program is provided in a moulded plastic video cassette-type box, which contains two disks (one is a back-up), a story book, a book of ideas for the teacher, a user's manual, an audio cassette tape containing music and a dragon story, a cardboard dragon template, and the names of the game's various treasures on small cardboard pieces. In order to be able to operate this version of the program, a single disk drive is essential.

Optional hardware involved includes a cassette player to play the audio cassette, although this has no direct relationship to the program – it does not need to be played while using the computer, as it contains dragon 'theme' music and a dragon story. A printer may also be used to print out a copy of the riddles used in the program, and to do a screen dump (print the current screen) of a maze to the printer. A colour monitor is essential in order to be able to complete certain sections of the adventure which rely on colour cues. Since this revue was carried out on a green screen monitor, and most school computers do not have colour monitors, this is a serious drawback when using the program.

The software itself is comprised of three parts. The first is The Adventure Part 1, the second is The Adventure Part 2, and the third is a utilities section which allows the use of the keyboard to play music, to solve and print riddles, and to solve and print mazes.

The objective of the student in Adventure Part 1 is to arrive in *Dragon World* by finding the five magical teeth of 'Bewgo'. The first tooth is to be found by counting the number of musical notes played by a dragon and typing the number when asked. There is no way out of this section other than to answer correctly. If

the student is unable to, then the only way out is to restart the computer. Once the correct number of notes played has been entered, a riddle is asked. If the student answers 'dragon' another riddle sequence follows. In order to move on from this sequence, the student must find the correct answer to the riddle and thus gain the first of the magic teeth. If the student answers 'egg', eighteen eggs are arrayed on the screen and the student's task is to find the second hidden tooth behind one of the eggs, and then to find the baby dragon's egg. Once this is found the dragon emerges from the egg and requires feeding. Only dragon food is acceptable – it must begin with the letters in the word 'dragon' – and the weight of the dragon has to reach 100 kg. When it does, another riddle will be asked allowing access to the next part of the adventure through three windows, at the same time as discovering the third tooth.

Window two should be chosen first as it gives access to six sorcerers, some of which can help find the next two teeth. Three have to be chosen to accompany the student. In window one behind two brick walls, resides the juggler. Provided the student has chosen helpful sorcerers and has worked out how they can help (the sorcerer Regdab helps to go under the wall, for instance, since his name is 'badger' spelt backwards), the two walls are breached, and the ball dropped by the juggler is caught. This gains the fourth tooth. If the walls cannot be breached, or if the ball cannot be caught, the fourth tooth cannot be found. If the student gets through the walls, but cannot catch the dropped ball, then the student is stuck at this place unless the machine is restarted. In window three there is an invisible randomly drawn path which the student has to follow by using the up, right, and left arrow keys. If the sorcerer Nogard has been chosen, he will provide musical notes to tell the student if the chosen path is correct, too far to the left, or too far to the right. The student is given two attempts to follow the path and find the last tooth. If that is done, the student enters Dragon World and is given the password necessary to

enter Part 2 of the adventure. If any teeth have not been found, then a clue is given to help the next time through.

The Adventure Part 2 involves finding five treasures which are to be presented to the dragons. There are twenty nine different treasure-bearing locations spread throughout five different sites. The procedure for discovering these treasures is similar to that used in Part 1, namely, with the assistance of the sorcerers who are now animals. If an animal has helped once, it cannot be used again unless it is fed with appropriate food which can also be found. Only the dragon can be used without further feeding. When five treasures have been found, the adventure can be ended. The ultimate ending in terms of treasures found, is to locate the 'real' treasures rather than the commodity treasures. These 'real' treasures are: friends; peace; kindness; health; warmth; laughter; love; and happiness.

Content Evaluation

The program documentation declares the objectives of the program as follows:

> *Dragon World* was not designed to teach children anything. Its purpose was to provide a gateway to another world, a world in which children would be stimulated and motivated to ask questions, find answers, discuss issues, keep records and use their imaginations to make that world their own world. Above all its purpose was to give teachers an opportunity to make the classroom a good place to be in. (Matson, 1985, 4)

This objective is both worthwhile and relevant to the curriculum. The target audience is not defined at any stage, although the level of language used would make it appropriate to children of age eight years and above, while teenagers would probably find it all a little 'babyish', both in language and in theme. The content really comprises a story on a disk, with which the student can interact, and as such, the accuracy of the content is not relevant to this evaluation. Whether the content is

appropriate or not depends on the teachers and students who intend to use such a program and how it is integrated into their curriculum. Using the rating scale as shown on the form in Appendix A, the content of *Dragon World* is awarded three points.

Program Features Evaluation

Dragon World is fully error trapped and wrong key presses are ignored by the program. The presentation is a little confusing in that the program continually switches its progress key (you need to press it in order to go on to the next screen) between the **return** key and the **space bar**, when it would have been more straightforward to have simply used the **return** key throughout. Another annoying feature is that if students need to restart the program, they are forced to sit through a rather long and tedious introduction before they can begin the program itself. As the likelihood is that this will happen often, particularly when they are just beginning to use the program and are unlikely to solve the adventure, it can become very tedious and may stop students from using the program.

A major problem with the presentation is that there are several points in the program where it is possible to become stuck unless a solution can be found. There is no mechanism provided, such as helpful hints, for example, to assist the student. The only way out of such impasses is to restart the program. Again, a feature such as this will quickly stop students from using it as they give up in frustration. Another major problem is that the solution to several puzzles posed throughout *Dragon World* depends upon colour being available. One such is the crossing of Bewgo's Stream in Part 2. There are stepping stones across a stream which may be any one of five different colours. As the only way to cross the stream is to type the initial letter of the colour of the stone you wish to step to, this presents an impossible task for those without colour monitors. There is no other differentiation, such as different patterns, between the stones to help identify them

other than by colour. Since the majority of school users will not have this colour feature available to them, this is a major drawback to its use in those schools.

Another poor program feature occurs in Part 1, during the baby dragon feeding segment. The baby dragon must be fed with correct dragon food. This is identified by the fact that it begins with any one of the letters in the word 'dragon'. Once a food has been fed once, it cannot be fed again. The problem arises in the fact that the program only examines the first letter of the food entered before accepting or rejecting it. This means that absolute nonsense such as 'ninny', 'deadheads', or even 'rhoshflhf' is accepted as being food simply because of the first letter. If the same nonsense is entered again, the dragon responds with comments such as 'I've had enough deadheads'. It isn't difficult to imagine what might be entered as dragon food by some students.

A final feature which lessens the usefulness of the program is that the pathway through the adventure, particularly in Part 1, is always the same. This means that once a student has solved each part of the adventure, it is no longer useful, interesting or challenging for that student. It would have been relatively simple to design some random aspects into the program's story-line, so that when a student had reached a correct solution once, a different pathway through the story could have been followed, with slight variations on the original. The other problem with this aspect of the program is that students can get solutions from others who have solved the puzzles, thus lessening the impact of the objectives expressed earlier. Using the rating scale as shown on the form in Appendix A, the program features of *Dragon World* are awarded two points.

Special Features Evaluation

The adventure itself has no special features associated with it. On the *Dragon World* disk, however are four supplementary programs. The first of these allows the student to use the computer keyboard to play notes, the aim being to allow

students to experiment with different sounds. Whether this is desirable in a classroom, or even in a computer lab will have to be decided by the teacher. This feature can also be used to save music onto a disk, to listen to music already saved on disk, and to listen to a random selection of notes and then select which is the highest or the lowest. After a selection is made, the student is advised whether the choice was right or wrong, a staff is drawn on the screen and while the notes are played, the notes are drawn on the staff. This feature could be used as part of a listening lesson, for example.

The second supplementary program is the Riddle Solver which allows students to try to solve the riddles used during the Adventure itself. The third supplementary program allows these riddles to be printed out, and the fourth program allows the student to practise moving through mazes, one of which is used in the Adventure, by using the arrow keys. The practice mazes are drawn randomly by the program, but each has only one solution. After each turn there is the option provided to repeat the same maze or to try a different one. Using the rating scale as shown on the form in Appendix A, the special features of *Dragon World* are awarded three points.

Classroom Application Evaluation

Although it is quite possible for the student to use the program entirely without any introduction, the idea behind the Adventure will be better understood if the teacher reads the Dragon World story to the students first, or plays the audio cassette of the story. This provides the background to the Adventure itself. The program can easily be used independently by most students either working alone, or in small groups. If students working alone find the program difficult, then the teacher will be needed much of the time to provide assistance. This is especially true of those times during the Adventure that the student becomes 'stuck' and cannot continue without assistance. Bearing in mind the objective of the program,

the classroom application of *Dragon World* is awarded four points using the rating scale shown on the form in Appendix A.

Support Material Evaluation

Except for one or two minor errors, the *Users' Manual* instructions are both adequate and accurate. Perhaps the major strength of the *Dragon World* package is *The Book of Ideas* which is provided with the program. This booklet provides a host of ideas and activities which can be carried out away from the computer based around the dragon theme. Topics covered include:

- Organising a *Dragon World* project;
- Art activities;
- Cross-curricular activities;
- Dance/drama activities;
- Environment activities;
- Language activities;
- Logic activities;
- Maths activities;
- Music activities;
- P. E. activities;
- Science activities;
- Dance/drama and music notes about the music on the audio cassette provided;
- A book list of stories about dragons and riddles; and
- A list of motivating theme music.

The more than one hundred ideas provided, are an excellent starting point for theme work on this topic. A matrix is also provided which illustrates how these ideas fit into the two parts of the *Dragon World* adventure. Using the rating scale as shown

on the form in Appendix A, the support materials of *Dragon World* are awarded five points.

Summary

This gives a total of seventeen points out of a possible twenty five to this software package, making it one that, despite its shortcomings, is a good package overall and should be considered for purchase.

Clearly, the evaluations given above reflect the opinions of one evaluator only. A well-rounded view of the software could be gained by comparing the reviews of several such evaluations, each carried out by a different evaluator.

VII
SOFTWARE DESIGN

Perhaps the single most important software design element is that the program you wish to develop must be designed specifically to meet student needs. A program designed to teach young children numeracy skills, will differ greatly from a program designed to teach numeracy skills to innumerate adults. In general, there are four basic characteristics that make up a quality software program. These are; clarity, simplicity, attractiveness and motivational appeal, and efficiency. This chapter examines how best to implement these characteristics when designing software.

CLARITY

The way in which a message is phrased or presented can have variable effects upon student performance. Close attention must be paid to the style of the message, to the intended user, and to the intention of the message. It is better to make use of simple, concise sentences rather than long ones with complex wording. Present only one idea per frame/screen, and keep the message clear both in language and graphics. Use plenty of spacing and preferably leave a blank line on screen between each text line.

Style

It is vital that all spelling, grammar, punctuation and usage is correct. This applies to all aspects of the software package, including the support material. It is wise to have someone other than the original author(s) check these items as a final

proof, since it is very easy to overlook an obvious error. There are implications here for software that is intended for an international market, especially in the areas of spelling and usage. It is wise to avoid jargon or slang terms as they tend to date very quickly, they generally do not travel very well (especially internationally), and younger students may not have the same understanding of the terms that the author has in mind.

Be particularly careful with humour, as what is funny to one person may not be to another, and what is funny to an adult may not be for a child and vice versa. While humour may be appealing at times, there is no evidence to support the notion that humour enhances information acquisition. Particularly avoid any reference that may indicate bias or discrimination in the areas of race, sex, creed, colour, religion, national origin, age, marital status, and physical and mental handicap.

Always try to communicate with the user in a friendly manner. Use a conversational tone, as many students attribute a personality to the computer and computer programs. Use first person, and where possible, use the student's name. As a general rule, create the program in such a way that it 'talks' to the student as one would in person.

Intended User

Ensure that the vocabulary used is at a level appropriate to the intended user. New words may need to be defined. This may be done graphically on screen, or students may be able to access a Dictionary of terms used, or definitions may be provided in the support materials.

It is desirable to present only one idea or instruction on the screen at any one time. There is no need to fill the screen with text as if it were a book. Because the student needs sufficient time to read and respond to text, either check that the speed of text presentation is appropriate to the student's level or, preferably, allow the

student to go on to the next item when ready. Since there can be a wide range of reading and comprehending abilities among students using a program, it is better to give the student control, such as by pressing a particular key to continue. As students are used to the page turning approach of text presentation, it is preferable to use this method in programs, rather than scroll the text up the screen. Use of this method also reinforces the left-to-right top-to-bottom skills used in reading texts.

Ensure that graphics presentation is suited to the level of the intended user. Enlarged text can be used for younger students, and should be used in programs for the visually impaired. If possible, an auditory presentation of the text is also a good idea.

Language that is graphical in content and style is likely to be remembered better than abstract language. Images are often easier to remember than words. As well, younger students find it easier to grasp concrete ideas rather than abstract ones (as discussed in Chapter IV). Consider the environment in which a child learns a language. The concept of a 'dog' is presented in the form of a real dog, a picture of a dog, the word 'dog', the sound a dog makes, and all other characteristics than combine to forge the concept 'dog'. By providing a multiplicity of sensual input, there is likely to be developed a greater understanding of the concept of a dog. It is easy to make use of all these characteristics in a computer program, short of actually producing a dog.

Message Intention

While messages, or 'feedback', are used to reinforce correct responses, the primary function of feedback should be to locate errors and to provide information about those errors to the student, so that they may be overcome. Providing positive feedback is less facilitative during acquisition of instruction than giving negative feedback. Feedback should, therefore, be used primarily after wrong responses.

Feedback is not necessary after most correct responses. As has been seen earlier in this study, the behaviouralists (such as Skinner) have demonstrated that an intermittent reinforcement schedule maintains learning for long periods. Of course, the computer is an ideal machine for using the progressive reinforcement schedule as described in Chapter III.

The messages used for positive feedback should be varied and random. In other words, positive feedback messages should be drawn at random from a database of appropriate words, phrases and sentences. Feedback should be non-threatening and thus 'user friendly'. Take a positive approach to feedback, using messages such as "Nice try (name), but you're not quite right. Have another go." rather than "No, you're wrong! Do it again!" Feedback should be immediate and should inform the student of the correctness or otherwise of the response. The author should also attempt to anticipate all possible student errors in order to be able to provide appropriate helpful feedback. If, for instance, the response '10' was received for the question '6 x 4', the feedback provided could be 'You have added instead of multiplying. Try again.' or 'The problem is 6 x 4, not 6 + 4. Try again.' or some other meaningful response. The point is that such responses need to be anticipated and allowance made for them. As an example of how complex this task can be, refer to Appendix B in which Brown and Burton (1978) examine common procedural 'bugs' in basic subtraction exercises.

When judging student responses prior to offering feedback, ensure that the program accepts as correct all reasonable synonyms, equalities, or word variations unless exact responses are required. There is nothing more frustrating than being told by a computer that a correct answer is incorrect, simply because the author forgot to include it as a possibility.

SIMPLICITY

It is most desirable that students can exit from an activity and enter a new one, if so desired, just as a pages in a book can skipped, chapters ignored and so on. So a program should have the capability of allowing students to exit easily at any time. This is important in other respects as well. A 'module' duration of 15 to 20 minutes would seem to be the optimum time. The more the task demands mentally, the more rest is needed to keep performance at an optimum level. If the program requires a longer running time, allow the opportunity to exit, and, if necessary, save the lesson status to disk so that the student can re-enter at the previous exit position.

What is expected must always be obvious to the student. If this is not so, then an explanation should be provided on screen. If a student is unable to answer correctly or to carry out a required operation, there must be a means provided whereby that student can continue using the program, whether this is simply to go on to the next step, or to be re-directed to some other section of the program, or some other course of action. The student must never be left in a situation where the only alternative to entering the correct or expected response is to stop the program altogether.

Avoid unnatural codes or response keys. For instance, if the answer to a question is 'Yes' or 'No', have the student respond with 'Yes' or 'No', but not '1' or '2'. Most students will also appreciate being able to press 'Y' for 'Yes' and 'N' for 'No', rather than being forced to type the whole response unnecessarily. In order to avoid keyboard confusion, it is better to display upper-case characters for those who are unfamiliar with the keyboard. For example, a display should read 'Press A, B, or C' rather than 'Press a, b, or c'. Once a response code has been established, use it consistently throughout the program, and, indeed, throughout all programs. Another consideration to be aware of is the need to ensure that keys

assigned as functions conform to the conventions in place for that particular computer. In most instances, the student should be aware of all active keys that can be used in the program.

ATTRACTIVENESS AND MOTIVATIONAL APPEAL

Although we process information with all our senses, vision is the major channel of input from the computer. The effective use of colour can enhance educational software (Durrett, 1982). Researchers have found that when comparing colour and black and white audio-visual presentations, there was a greater incidence of reminiscence in colour over black and white presentations after a period of one week (Farley, 1976). Others have found that the use of colour facilitated recall of lists of words whether they were grouped together or arrayed randomly (Elio, 1978).

Colour can be used to code, differentiate or highlight instruction. Colour can direct attention to various aspects of the screen display. For example, special keys for the student to use can be colour highlighted, such as Help, Calculator, Menu, and so on. Colour patterns can be based on the colour hue (red, green, blue), the location of the colour display on the screen, or on the function a colour performs. When using colour based on a function, try to make use of them as they are used habitually: red for stop and green for go, for example.

It is important to be aware that even though a program may be designed in colour, not all users will have a colour monitor. It is therefore necessary that the display works effectively on various other monitors, such as green, amber or black and white, as well as on normal colour television sets which can be used as monitors with some computers. Additional cues, such as underlining or shape cues, may be needed to emphasise key points in the text and to compensate for the

lack of colour. Remember also that some students are colour blind and will always need additional cues.

It should also be realised that not all users will have high quality colour monitors. By avoiding red/green, blue/yellow, green/blue, and red/blue pairs, and by making use mainly of the primary hues of red, green and blue you can help avoid colour quality problems. Readability will be improved by using high colour contrast for character and background pairs. Dark colours on a dark background are harder to read than light colours on a dark background. Alpha-numeric characters are best in red, white or yellow, while light blue should be restricted to large background areas only. For most users, the number of colours used for each display screen should be restricted to four. As the number of colours increases, increase the size of the colour-coded object (Durrett, 1982).

Sound can also be used to enhance a program. It can typically be used as a reward for a correct response, or to draw attention to incorrect response or errors. Care needs to be taken that an error sound is not so attractive that the student keeps making errors simply to hear the sound, thus detracting from the positive learning aspects of the program. As there will be many instances where computer sounds may be an unwarranted distraction, such as in the case of a computer being used in the classroom while other lessons are being conducted, it is important that there be a facility within the program that enables sound to be turned off. Some computers have earphone sockets which may be used for this purpose.

Graphics, likewise, can add to the attractiveness of a program and to its instructional appeal. Once again, however, care needs to be taken that graphics do not merely provide a distraction or an unwarranted delay, such as repeating the same animation after every response. This will very quickly cause student frustration or boredom, and subsequent loss of motivation. Animations used as rewards, for instance, need to occur at random intervals, and to be of as many

different types as possible. Both graphics and sound should be appropriate for the intended user. When using graphics and animation, it may at times be necessary to to draw attention to certain areas of the screen by using underlining, flashing, or colour.

Authors need to be aware that both graphics (animation in particular) and sound are usually very memory expensive. It would therefore be wise to consult with a programmer on the machine types for which you are developing software, so that you are aware of the limitations of the hardware.

EFFICIENCY

Design a program keeping in mind the idea that the computer/pupil relationship works best as an interactive one. If you work closely with your programmer, you will become aware of instances where delays will occur. At such times, let the student know that the delay is normal by presenting an appropriate message on screen such as: "Please Wait"; "One moment please"; "Writing the file to disk"; "Printing in progress" and so on. If student progress is being monitored by the program, such as in a Test, inform the student of that fact. Any data collected and stored should be protected from the eyes of anyone other than the student and the instructor by means of a password, for example. Data should only be able to be erased or altered by authorised persons, controlled by the instructor.

When the program has been coded, it is necessary to check the efficiency of its operation. Check that screen displays always clear properly and do not overwrite one another, so that the user may see half of two sets of screens. Try to ensue that screens are displayed as quickly as possible in order to avoid unnecessary delays during the operation of the program. If this is not the case, you may need to explain to your programmer the need for maximum speed in such cases, and more efficient coding may be possible. Check that the information

displayed on the screen is visually balanced. If you do not have a good sense of design, seek the opinion of a graphic designer or artist. Be consistent in the placement of displays. For example, the bottom text line of every display may always contains information on how to return to the startup menu.

It is also necessary to check the operation of the program for durability and efficiency. Make sure that the program is totally error-free in operation, that it starts up easily as it should without errors and carries out any other function as it should – printing, for example. When the program is operating, run some tests to see how it copes with unexpected input or responses. What happens, for instance, if, when the expected response is A, B, or C, the D key is pressed or if a number is pressed? Make sure that the student cannot get stuck in an infinite loop with no way of escape, such as in the case of an incorrect response. Check that all special keys function as they should and when they should. Is Help available at any time, for example? Can the program be 'crashed' either accidently or deliberately by erroneous or mischievous use of the keyboard? In general, test every conceivable aspect of the program you can to see whether or not the program will cope. When you think all is well, then have the program field tested in a situation such as that in which the program will be used. Ensure that any problems are remedied before releasing the program for general use.

SUPPORT MATERIALS

Software programs may be enhanced by the provision of clear, useful support material to which students and teachers can refer when necessary. Examples of such materials are charts, maps, stencil masters, tables, equations, figures, diagrams, operating instructions, users' guides, instructors' guides, reward stickers, and so on. The use of some of these materials can lower computer processing capacity demands. This becomes a necessity at times due to the lack of

memory or to the lack of disk space available. Although it may be preferable to have all information on line, it is not always possible to do so and still remain within the computer's operational limitations.

VIII
SOFTWARE DESIGN: A CASE STUDY

By the application of the foregoing information, it is now possible to examine the construction of a prototypical software package which should make the best possible use of that information. The software package to be examined is *Number Detective* (Greig & Stace, 1985e), which was co-designed by Ian Greig and Raymond Stace with assistance from Peter Burton and was programmed primarily by Michael Fazzolare, with assistance from Michael Shepanski and John Solyom.

DEFINING THE PROBLEM

The correct conceptualisation of number theory at an early age is of vital importance to children's development in mathematics throughout their academic careers. As Inhelder states:

> One wonders... whether it might not be interesting to devote the first two years of school to a series of exercises in manipulating, classifying, and ordering objects in ways that highlight basic operations of logical addition, multiplication, inclusion, serial ordering, and the like. For surely these logical operations are the basis of more specific operations and concepts of all mathematics and science. It may indeed be the case that such an early science and mathematics "pre-curriculum" might go a long way toward building up in the child the kind of intuitive and more inductive understanding that could be given an embodiment later in formal courses in mathematics and science. The effect of such an approach would be... to put more continuity into science and mathematics and also to give the child a much better and firmer comprehension of the concepts which, unless he has this early foundation, he will mouth later without being able to use them in any effective way. (In Bruner, 1974a, 420)

Many approaches have been tried, with varying success, to find methods of assisting children to correctly visualise, interpret, and name number concepts. If this can be successfully achieved then, "... erroneous interpretations and images that can hinder children's development in mathematics will diminish, and bugs in

arithmetic procedures will disappear. Learning mathematics will be more enjoyable." (Greig & Stace, 1985e, 12) The computer environment, and a microworld environment in particular, seem to be ideal for the introduction of such concepts.

DESIGNING A MICROWORLD PROGRAM

A microworld environment is one in which the students are free to explore and discover powerful ideas and concepts for themselves.

> Learners in a ... microworld are able to invent their own personal assumptions about the microworld and its laws and are able to make them come true. They can shape the reality in which they will work... they can modify it and build alternatives. This is an effective way to learn, paralleling the way each of us once did some of our most effective learning. (Papert, 1980, 126)

Objects in a microworld are able to be transformed, and that transformation can later be reversed (Adams, 1986). These transformations are modular so that they can easily be decomposed into chunks and are made according to a set of rules which the student will be able to discover. By exploring such microworlds, students " ... learn to transfer habits of exploration from their personal lives to the formal domain of scientific theory construction." (Papert, 1980, 117) The *Number Detective* microworld provides a structured, visually stimulating environment for self-guided discovery learning, in which students are encouraged to discover aspects of number theory for themselves.

OBJECTIVES OF *NUMBER DETECTIVE*

In order to be able to design such a microworld, it is first necessary to define a set of mathematics objectives which are achievable through use of the program. The objectives selected were defined after reference was made to the *Curriculum for Primary School Mathematics* (1972), and to Pirie and Jecks (1970, 1971). For the *Number Detective* program, children are provided with an environment in which they can:

- Name the shapes in a set.
- Discover the numerals to twenty.
- Discover that the numeral 0 (nought) is used to indicate a set that has no members.
- Discover that the numbers to twenty can be represented by concrete shapes.
- Count to twenty by ones with understanding.
- Read and understand any numeral to twenty.
- Discover that every set has a cardinal number.
- Discover that take away is the inverse of add and that add is the inverse of take away.
- Discover that a set's cardinal number remains the same even if the objects in the set are rearranged.
- Discover that a cardinal number may be represented by either a numeral or a set.
- Discover that a set is counted by pairing its objects with the names and the numerals of the cardinal numbers taken in counting order, such that the last name used is the name of the cardinal number of the set.
- Find a particular object in an ordered set when given its ordinal name.
- Discover that sets may be partitioned into disjoint subsets, each containing a different set of objects.

- Sort objects in a particular set.
- Select items from a set which have some common feature.
- State which objects in a set have common features, and identify objects which do not share features with the common-feature subset.
- Find a set equivalent to a given set of ten or fewer objects.
- Combine objects with common features into a subset, or combine two such subsets.
- Discover that when items are grouped into fixed size subsets, in some cases all items will be grouped, while in other cases some items remain ungrouped.

(Greig & Stace, 1985e, Reference Manual to *Number Detective.*)

PROCEDURE DEFINITION

Having defined the objectives, there now needs to be defined a set of procedures which will enable the child to achieve these objectives when using the program.

Table 1 Operational procedures for the *Number Detective* program

Procedure Name	**Procedure Operator**	**Procedure Effect**
Add	The **A** key	When the **A** key is pressed, a shape randomly chosen from a shape bank is Added to the line of shapes on the screen. The first shape is located at the top left corner of the screen. The program is divided into three levels. In Level One, up to five shapes can be Added, up to ten in Level two, and up to twenty in Level three.
Take Away	The **T** key	When the **T** key is pressed, one shape is Taken Away from the line of shapes.

Count	The **C** key	When the **C** key is pressed, the shapes in the line are Counted one by one, each pressing of the **C** key accounting for one shape.
Scatter	The **S** key	When the **S** key is pressed, the shapes are removed from the line and randomly Scattered around the screen.
Line Up	The **L** key	When the **L** key is pressed, the shapes are Lined Up and Counted automatically.
Mark	The **M** key	When the **M** key is pressed, a selected shape in the line is Marked by having a line drawn under it
Gather	The **G** key	When the **G** key is pressed, the Marked shapes are gathered together at the front of the line of shapes.
Join	The **J** key	When the **J** key is pressed, selected shapes are Joined together by having a frame drawn around them.
Zap	The **Z** key	When the **Z** key is pressed, Counting, Marking or Joining lines are Zapped or removed, and the procedure is ended.
Box	The **B** key	When the **B** key is pressed followed by a number from 1 to 20 and the **return** key, the shapes on the screen are boxed in groups according to the number typed. If the **B** key is pressed, followed by the **3** key and then the **return** key, the shapes on screen are boxed in groups of three.
Help	The **H** key	When the **H** key is pressed, a screen is displayed which names the functions of all the keys used in the program, including how to restart the program.
Restart	The **Del** key	When the **Del** key is pressed, the program is restarted, allowing the student to choose a different level from the current one. This key was chosen because it is away from the other keys on the keyboard and thus should not be accidently pressed.

PROCEDURE IMPLEMENTATION, and ENVIRONMENTAL DESIGN AND CONTROL

As this program is intended, in the first instance, to be used by children as young as four years of age, the input mechanism needs to be as simple as possible. Therefore, the execution of each command for levels one and two of the program is achieved by a single keystroke only. Further, the commands available at any given time are always displayed (in white for clarity) in the bottom right hand corner of the screen and the command just implemented (the key just pressed) is always displayed on the same line in the bottom left hand corner of the screen. If a non-functional key is pressed, the speaker beeps once to indicate that this is so, and the functional keys available to be pressed are slowly flashed in order to indicate that the student should try one of those. As each procedure used in the program has its own unique accompanying sound, and as there will be times when the sound needs to be turned off, this can be done by using the **Control** and **P** keys, the letter **P** representing Play the sounds. Pressing the same key combination turns the sound back on again. The program is designed to be operated in full colour, but is not limited in any way by being used on monochrome monitors.

After the title screen has been displayed, the first screen the student sees asks for the Level to be attempted, as shown in Figure 8·1.

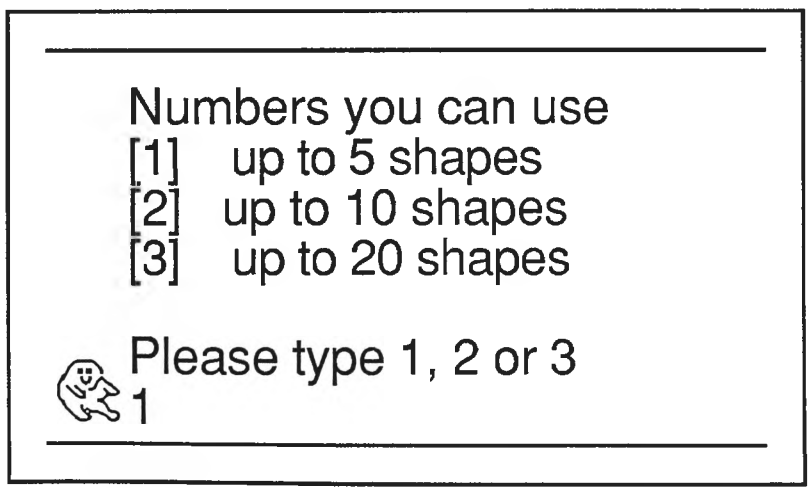


Figure 8·1 Entry level choice screen

All the student has to do is type either 1, 2 or 3 in order to begin the program. The number appears as shown above after it has been typed and the program moves on to the next screen, which appears as follows, in Figure 8·2.

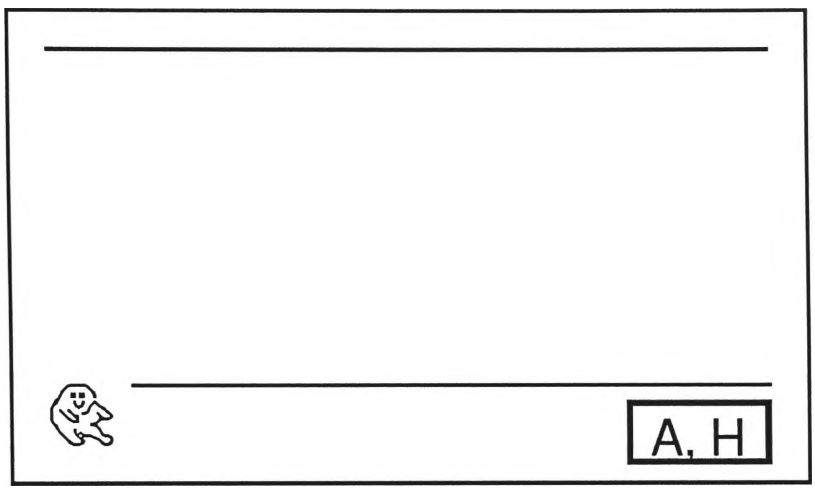


Figure 8·2 Work initiating screen

The letters **A** and **H** which are shown are the only commands available at this stage, and they are accessed by pressing those keys on the keyboard. The concept behind the program is that of discovery learning. The aim was to structure the program with such small bits of information that they would be the building blocks for knowledge and concepts, but they would also be 'pre-conceptual' facts. The learner would then be left with a sufficiently high ceiling in terms of conceptual complexity to be able, by discovery, to work at any given level of concept within the parameters of the program. The program is thoroughly error trapped, so that if a non-functional key is pressed, the program cannot be accidently interrupted. The philosophy of the program is that the student may experiment with many keys on the keyboard, finding that nothing happens until either the **A** key or the **H** key is pressed. The parent or teacher may, if he or she wishes, explain that the letter **A** shown on the screen corresponds to the same letter on the keyboard. Coloured stickers may be attached to the keys used by the program if desired. During personal discussions, King pointed out that the advantage gained by taking this

approach is that such a procedure would serve the purpose of multiple cueing in the sensory domain for young children for whom rapid and satisfying familiarity with the hardware could prove necessary to provide motivation to continue. This approach was left optional, as Greig and Stace objected that such a procedure, if prolonged, would defeat a secondary objective of the program, namely one of computer literacy and keyboard familiarity. It is also difficult to provide a quick and durable solution which does not interfere with the usage of the normal keyboard by other children who don't need such assistance, and which does not interfere with the use of the keyboard for other programs. A point made by Forwood (1986, 122) when using LOGO in a Kindergarten classroom is that:

> One interesting aspect of learning style arose by chance. During one teacher directed session, some of the computers had been left without coloured stickers on the keys. Some of the more outgoing students expressed a preference for this situation, claiming they wanted to decide what keys to press. Thereafter, some computers were always left without coloured stickers.

Students should preferably only be shown enough to allow them to begin using the program. They should then be left to discover for themselves as much as they can. Teachers and parents should not expect that the child will fully understand all the concepts and features presented in *Number Detective* during the first few sessions. Indeed, the program is intended to be used over a time span of as long as two or more years, depending upon the individual differences of each user. The bright five year old may take a shorter time successfully to attain the concepts than a remedial seven year old, for instance. The range of shapes on the screen can be used by the teacher or parent as a guide to the number of different commands which have been used by the student in any session. Initially, the number of different shapes available from the shape data bank at the commencement of each session is two in levels one and two, and three in level three. As the student uses more commands, more shapes become available for each subsequent use of the Add procedure.

Add, Take Away

If the **A** key is pressed, a shape is Added to the top left of the screen. This position was chosen to reinforce the top-to-bottom, left-to-right skills required for many learning tasks, but particularly for reading. The resulting display takes advantage of the fact that "... by the time they enter school most children have already constructed a representation of number that can be appropriately characterized as a mental number line." (Resnick, 1983, 111) At the same time the cardinal number of the set is shown using a large white numeral in the bottom right of the screen. This is done as subitizing pre-dates counting as a means of determining numerosity (Fuson & Hall, 1983). Subitizing is the ability to quickly produce the cardinal word for small sets of objects. Displaying the numeral assists the development of the subitizing skill. Children aged 5 years can subitize sets of one, two, and three (Klahr & Wallace, 1976; Resnick, 1983). If the **H** key is pressed. the Help screen is displayed. Throughout the program, the Help screen displays only those commands which are actually available at the time that the **H** key is pressed, in addition to "**Del - Restart**" and "**H - End help**", which are always displayed on the Help screen. Since **Del** is only shown on the Help screen, it follows that it only operates from that screen, since all functional keys are always displayed whenever they are available. In this instance it displays "**A - Add**". If the **A** key is now pressed, the screen appears as in Figure 8·3.

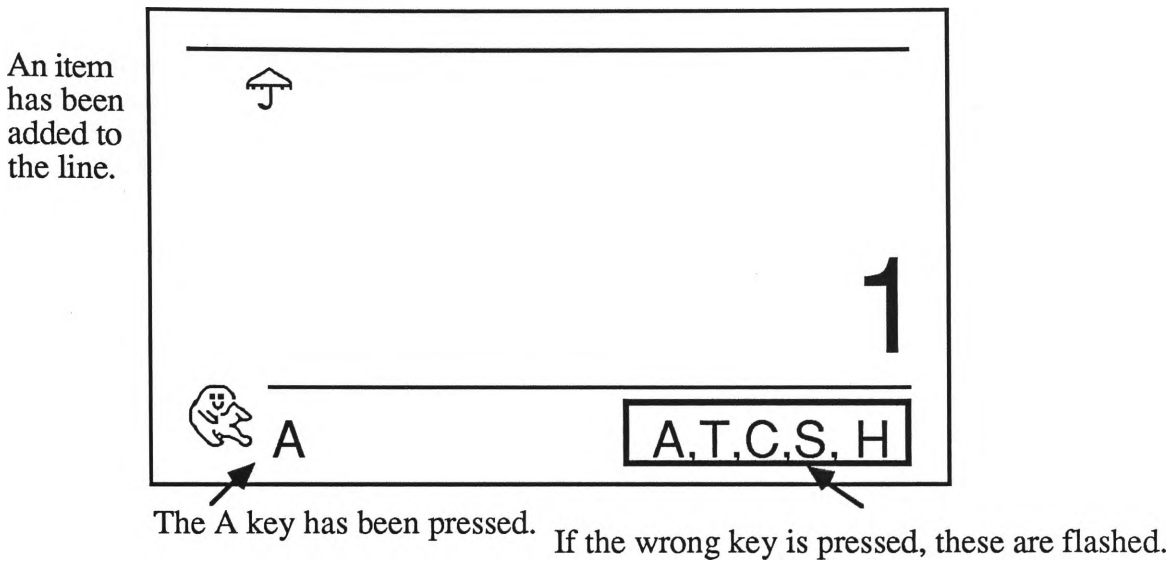


Figure 8·3 Add To, Take Away screen 1

Each time the **A** key is pressed and an item is Added to the line, a sound is played. When another item is Added, the sound is played again at a slightly higher pitch. The result is an ascending scale as items are Added. As items are Taken Away, there is a descending scale effect. This effect is used to reinforce aurally the concept that adding increases while taking away decreases the number of objects displayed. As there is now an item on the screen, it can be Added to, Taken Away, Counted, or Scattered, so these commands are now displayed. The Help screen, if accessed, displays these commands and their meanings, as just mentioned. If the **T** key is pressed, the item is Taken Away and the resulting screen would be the same as seen in Figure 8·2, except that the numeral 0 would be displayed for the first time. If the **A** key is pressed again, the screen appears as in Figure 8·4 below.

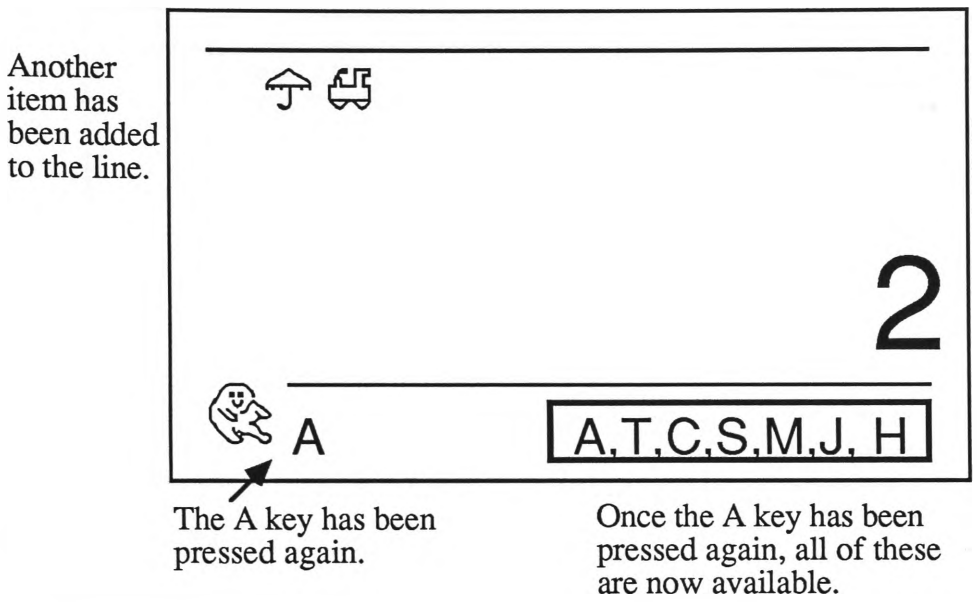


Figure 8·4 Add To, Take Away screen 2

As there are now two items, Mark and Join have become available for the first time. Mark allows the student to select wanted items and Mark them. Items thus Marked can then be Gathered together at the beginning of the line. This is illustrated in Figures 8·8 and 8·9. Join allows, in this instance, the two items to be Joined together into a single box. This is illustrated in Figure 8·11. If the **A** key is now pressed a further three times, one item is Added for each press of the **A** key, and the corresponding numeral is displayed in each case. When the **A** key has been pressed five times in total, the screen will be as shown in Figure 8·5. The shapes used in Figure 8·5 are for illustration purposes only, as any one of the twenty shapes available in the shape data bank (see Appendix C) could be chosen at random and appear on the screen instead of the ones shown here. If level 1 has been chosen at the first screen, then five is the maximum number of shapes that can be Added to the line. Therefore, the Add command is no longer available, until Take Away is used.

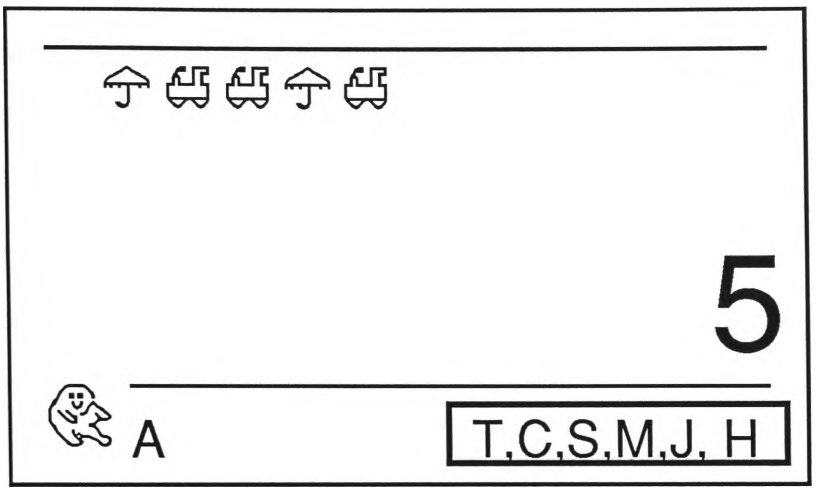


Figure 8·5 Add To, Take Away screen 3

Children will learn that Add and Take Away are the only commands which can change the quantity. "When children are observed to employ repetitively an [A], [T] or a [T], [A] sequence, then they are on the way to mastery of the inverse nature of Add and Take Away." (Greig & Stace, 1975e, 17)

Count

So far the commands Add and Take Away have been implemented. The next command available in the command box is **C** for Count. Fuson and Hall (1985) have found that children as young as 2 years can correctly produce the number word sequence "one, two, three", with no intentional one-to-one correspondence. By $3^1/_2$ to 4 years, the mean length of correct number word sequences has grown to 13. This means that children coming to *Number Detective* for the first time will probably already know the number words being used. When the **C** key is pressed for the first time, a Counting line is drawn on the screen under the first item in the line of objects. This emphasises the one-to-one principle of counting which says that each item in an array must receive one and only one counting word (Fuson & Hall, 1983) and assists with the internalisation of the pointing act which occurs with younger children when they begin counting, but is usually completely internalised by the age of 6 years. The numeral shown at the bottom right of the

screen changes to **1**, and the commands available are limited to **C** and **H**. In other words a Count, once begun, must be finished before other commands again become available. This emphasises the stable order principle of counting which says that the words used in counting must be produced in a fashion that is stable from trial to trial (Fuson & Hall, 1983). It also emphasises that the last number word said in counting a set of items gives the numerosity – the cardinal word – for that set. When the **C** key is pressed for the second time, the Counting line is extended so that it now sits under the first and second items in the line of objects, and the numeral at the bottom right of the screen is changed to **2**. This procedure is repeated until all four items have been Counted, when the screen in Figure 8·6 is displayed.

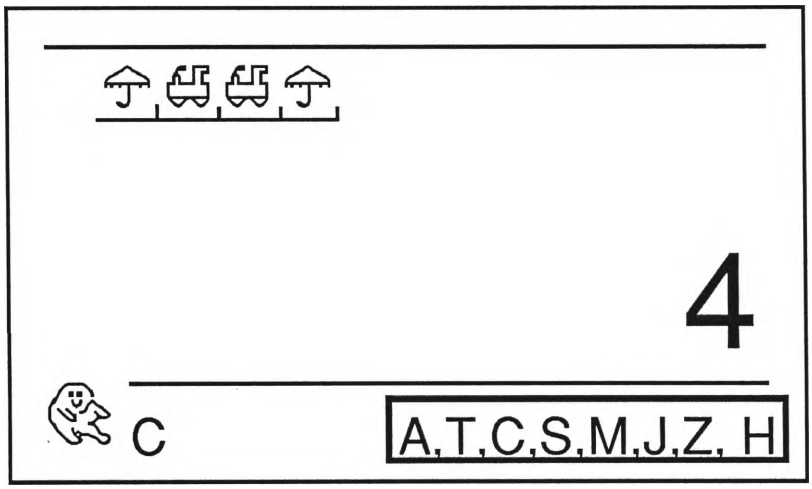


Figure 8·6 Count screen

It can be seen that the Zap command now becomes available for the first time. In this instance, pressing the **Z** key would remove the Counting lines from below the shapes. "When children repetitively press the [C] key just enough times to complete the count of a group of shapes, then the concept of Count is on the way to being mastered." (Greig & Stace, 1985e, 18) The cardinality principle has been acquired by the child when a correct response is given to the question 'How many?'. The common response of children who do not possess this principle is to

count the set again. Through extensive practice with counting the counting words are gradually transformed "... from a string of words into a representation of quantity in which each position (number name) in the list comes to stand for a quantity." (Resnick, 1983, 111)

Scatter, Line Up and Count

The next command in the command line is the **S** for Scatter command. When the **S** key is pressed, the items are removed from the screen and repositioned at random around the screen, as shown in Figure 8·7. Each time the **S** key is pressed, the same result occurs. To exit from the Scatter command, the **L** for Line Up command must be used.

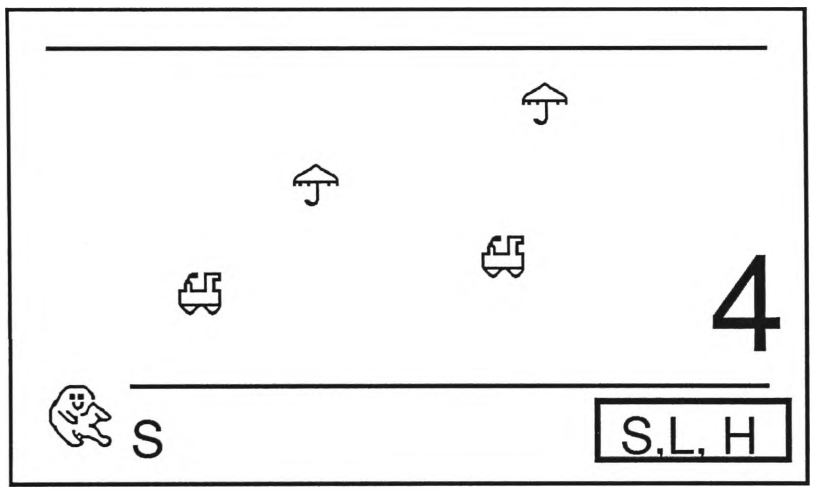


Figure 8·7 Scatter screen

When the shapes are Scattered, they momentarily disappear and reappear in different locations, as shown above. When the **L** key is pressed, the shapes stay on the screen and physically move back into a line at the top left of the screen. Different sounds are used for each procedure. Once they are in position, an automatic Count is made of the shapes, with the Count line being redrawn and with the numeral changing, to reinforce the fact that despite being moved all around the screen, there are still four items on the screen – indeed, if the student examines

them more closely, it can be realised that they are the same four items that were on screen before the Scattering took place. This procedure also emphasises the order irrelevance principal of counting (Fuson & Hall, 1983). Scattered shapes can be manually counted by students and the number they get can be cross-referenced with the number shown on screen. This exercise emphasises the abstraction principle of counting, which says that any collection of entities can be a set of countables (Fuson & Hall, 1983). "When children are observed to realise that the Scattered shapes are the same in quantity as the Lined up shapes, they are on the way to understanding the concept of conservation of number." (Greig & Stace, 1985e, 18)

Mark and Gather

The next command available in the command line is the Mark command. The aim of this command is to allow students to Mark and Gather together items of their choosing. The intention is that students will recognise that some objects in the line are like others in the line. Like objects can then be moved so that they are grouped together on the screen, rather than being spread throughout the line. When the **M** key is pressed for the first time, an arrow appears under the first object in the line. The student can then either press the **M** key again to Mark that object, or press the right arrow key to move to the object to the right. Moving along the line without Marking is achieved by using the arrow keys. In the sample screen shown below in Figure 8·8, the second and third objects have been Marked by pressing the **M** key when the arrow was sitting below that shape. The arrow then jumps to the next unMarked shape to the right, or if there is none, to the next unMarked shape to the left. If all shapes are Marked, the arrow disappears, and **M** and the arrow keys are no longer displayed in the commands available box.

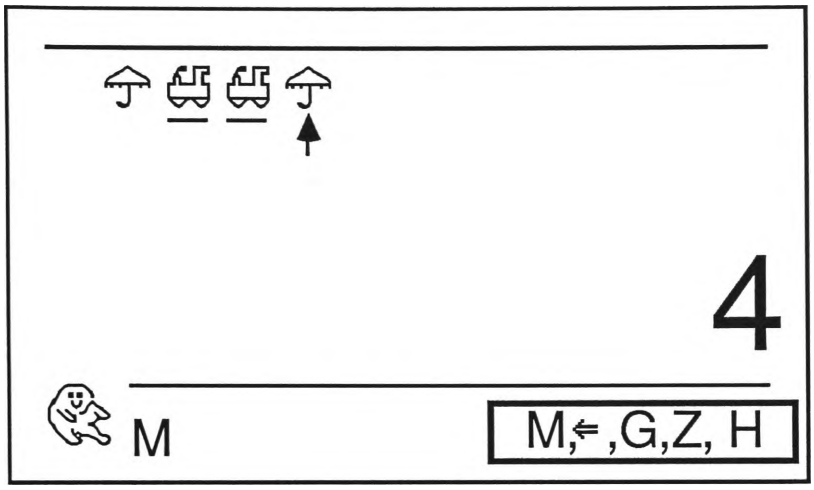


Figure 8·8 Mark and Gather screen 1

The next step in this sequence is to Gather the Marked objects together at the beginning of the line. This is achieved by animation. All shapes are moved down and around the screen to music, remaining visible at all times. They are then moved back into line in the order that they were Marked, so that Marked items are grouped together from the beginning of the line. This focuses attention on the left-to-right order of relevance. After Gathering the Marked items from the screen above, the screen shown in Figure 8·9 would result at the end of the animation.

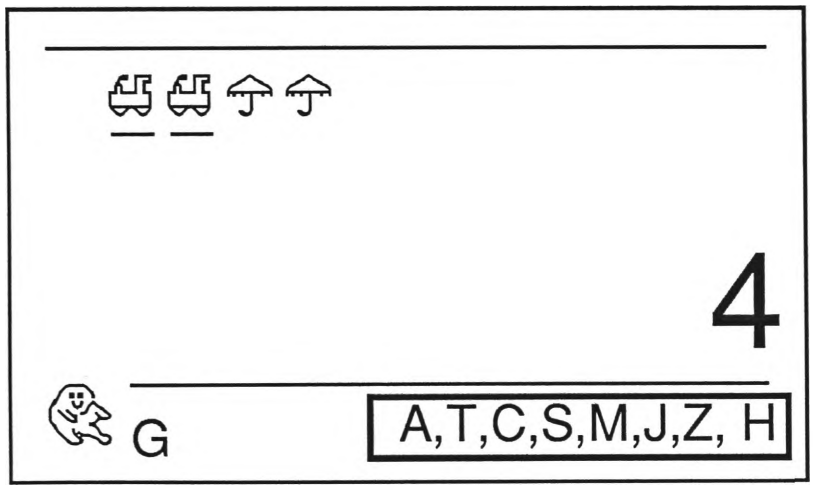


Figure 8·9 Mark and Gather screen 2

The Marked items are now situated together at the beginning of the line. In this case it coincidentally happens that the result is two pairs of like objects. The items remain Marked to reinforce the fact that they were the ones chosen before the

Gathering took place. They can be unMarked by pressing the **Z** key. The group of shapes can be partially or completely ordered using the Mark and Gather functions. Mark and Gather provide an introduction to the concept of a sub-group and thus to the whole-part schema of numbers which is fundamental to cardinality. "When children are observed to have Gathered shapes according to an intended pattern (such as like shape or colour), it may be thought that they have mastered the ordering commands [M] and [G], which complement the randomising commands [S] and [L]." (Greig & Stace, 1985e, 19)

Join

The next and final command available, at least for levels one and two, is the Join command. This command extends the introduction to the child of the part-whole schema of numbers which specifies that any quantity (the whole) can be partitioned (into the parts) as long as the combined parts neither exceed nor fall short of the whole. As Resnick (1983, 114) states:

> As long as the number line alone is used, there is no way to relate quantities to one another except as larger or smaller, further along or further back in the line. ... Probably the major conceptual achievement of the early school years is the interpretation of numbers in terms of part and whole relationships. With the application of a Part-Whole schema to quantity, it becomes possible for children to think about numbers as compositions of other numbers.

The Join procedure makes this possible in *Number Detective*. (See Appendix C, for some examples.)

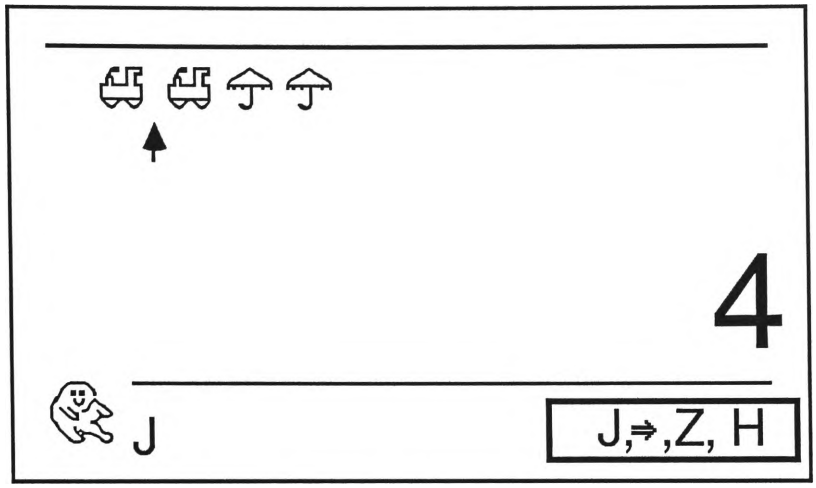


Figure 8·10 Join screen 1

The screen shown in Figure 8·10 illustrates what happens when the **J** key is pressed for the first time. The student can either press the **J** key again to Join the first two shapes together, or can press the right arrow key to move along the line. Both left and right arrow keys can be used to move along the line when such a move is possible. For instance, the student cannot move further right when the pointer is situated between the third and fourth shapes. If the **J** key is pressed again without further movement, the result would be as shown below in Figure 8·11.

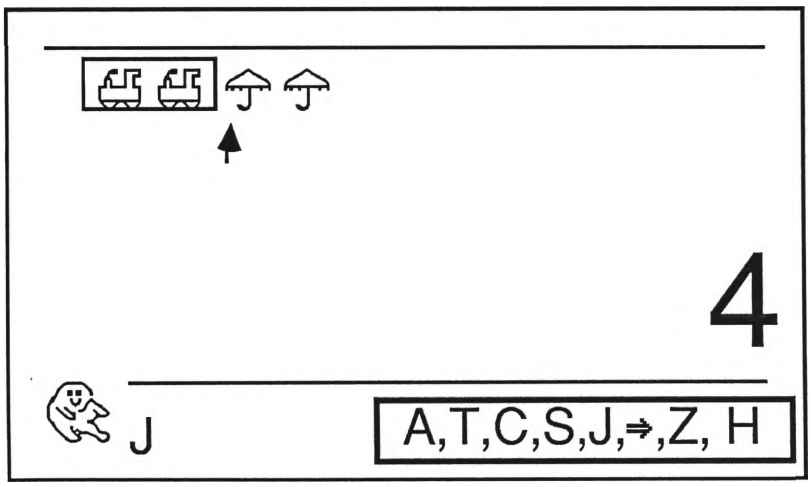


Figure 8.11 Join screen 2

The result of pressing the **J** key is that a box is drawn around the first two items, while the Joining arrow moves to point between the next pair. The box used in Join acts as an explicit sub-group identifier. This leads on to the establishment of

one-to-one correspondence between the elements of two or more sets and to the establishment of the relative numerosity of two or more sets. If the **J** key were now pressed again, the result would be a box containing three items with a single item left. All the commands available are again shown in the command box, including Count and Scatter. The Join box remains in effect until it is Zapped, whereupon the box is removed. The use of the Zap function demonstrates a conscious change in thought patterns from the use of cues or prompts from the previous procedure to a new procedure. The Count command functions slightly differently when some items are Joined, which provides an introduction to the idea that a group can be a countable unit. In the case illustrated above, the first two items would be Counted when the **C** key was pressed, followed by a single press of the **C** key for each of the remaining two shapes. This is done to highlight the fact that, in this case, there is a Group of two, and two single items making up a total of four. When items are Joined together, the Count line drawn is different from that used previously. It is now just a straight line (___) rather than the line with the raised section at the end as shown in Figure 8·6. If the Scatter command is used, the Joined items remain Joined throughout the Scatter, Line Up and automatic Count procedures. "When children are observed to have consciously explored all the possible sub-group variations of a line of shapes, they are on the way to mastering the concepts of grouping and sub-grouping, and are ready for the concept of addends." (Greig & Stace, 1985e, 19)

Box

It is expected that the usual path through *Number Detective* will be to complete level one (working with up to five shapes only), then level two (working with up to ten shapes only) and then to go on to level three (working with up to twenty shapes). As previously explained, all inputs to levels one and two require a single keystroke only. The same is true for level three, until the highest level of

command, namely Box, is reached. This new command is first introduced in level three and it introduces the idea that commands may have arguments (for example, those of the kind 'if this then do that'). When they do, a different input mechanism is required, and it is usually one which involves the use of the **Return** key. In the case of *Number Detective* this involves pressing the **B** key, then typing a number between 1 and 20, and then pressing the **Return** key. In the sample screen shown in Figure 8·12, the items have been Boxed into groups of three by pressing **B** then **3** then **Return**.

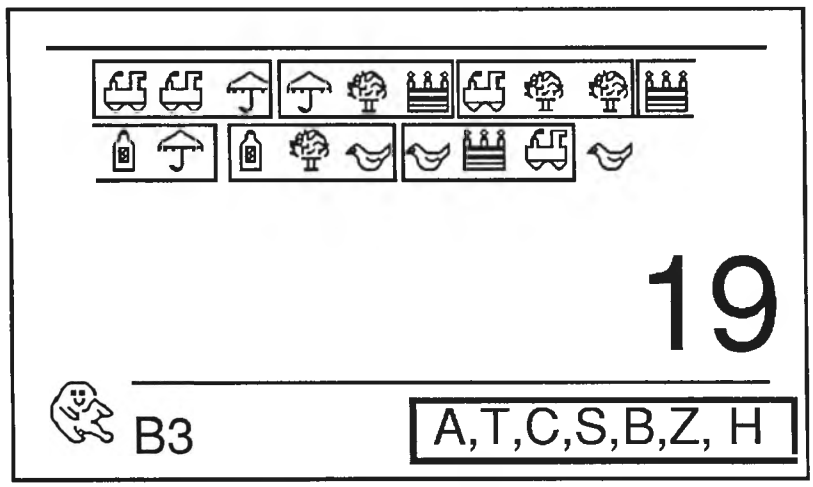


Figure 8·12 Factors and primes screen

The Box procedure provides an introduction to the base system of counting – up to base twenty – as it displays the line of shapes as equal sub-groups. The Box procedure provides cues for verbal statements about counting in any base. The Box procedure also provides an introduction to the concept of a remainder, as can be seen in Figure 8·12, factors and primes, and is a precursor to multiplication and division. As an example, the answer to the question 'How many threes are there in nineteen?' is illustrated in Figure 8·12. The student can easily count that there are six boxes and can readily see that there is one shape remaining unboxed. The Count procedure, when items are Boxed, works in the same manner as when items have been Joined. Similarly for the Scatter, Line Up and automatic Count

procedures. If two items were to be Taken Away from the screen shown in Figure 8·12, then the box would be removed before the eighteenth item was Taken Away. If one item was then Added to the seventeen which would be on screen at the time, a box would be drawn around the sixteenth, seventeenth and eighteenth shapes after that item was Added to the line. If **B5Return** was the next input, the current Boxes would be erased and new Boxes would be drawn around groups of five. "Children may be thought to have understood the Box procedure when they can describe numbers accurately in terms of equivalent sub-groups." (Greig & Stace, 1985e, 20) Children need to learn that if they count two sets and get the same final counting word for each set, then the two sets have the same number of objects. As stated by Fuson and Hall (1983, 75), "... there is now little doubt that a 'same counting word implies same cardinality' rule is one aspect of cardinal words that young children must learn."

SUPPLEMENTARY ACTIVITIES

There are many activities that the teacher or parent can use to supplement the concepts being examined in *Number Detective*. Some examples, which are provided in the *Number Detective* software package, are set out below.

(1) Count Aloud

At various stages through the program, ask children to count aloud the number of shapes on the screen. Try covering the counter number and asking how many shapes are on the screen. Encourage children to play outdoor games which involve counting, such as 'Hide-and-Go-Seek'.

(2) Name the Shapes

Ask children to name the shapes on the screen at various times.

(3) Shape Position

Ask children to identify which is the third shape, the sixth shape, and so on.

(4) Relative Positions

During the Scatter procedure, help develop both vocabulary and concepts by discussing the position of shapes on the screen. Use and ask for words such as near, beside, up, down, over, under, on top of, below, in the middle, and so on.

(5) Name the Colours

Ask children to name the colours on the screen at various times, provided a colour monitor is available.

(6) Match the Shapes

Ask children to identify like shapes, then use the Mark and Gather procedure to group them together.

(7) Match the Colours

Ask children to identify like colours, then use the Mark and Gather procedure to group them together.

(8) Add/Take Away

Discuss with children whether or not there are more than, less than, the same as, and so on during the Add and Take Away procedures. Also ask questions such as 'Is the line longer or shorter now than it was before?'

(9) Compare and Contrast

Discuss whether or not a number is more than or less than, larger than or smaller than another number, and whether one number comes before or after another number.

(10) Discuss

Talk with children about what they are doing with the program, and perhaps why they are trying particular keys.

There are countless additional activities, most of which are carried out daily in Infants classrooms, which can assist with learning the concepts involved in *Number Detective*. A few suggestions follow.

(1) Nursery Rhymes

Use Nursery Rhymes to help children learn and remember counting numbers.

(2) Write the Numerals

Have children write the numerals 1 to 5, 1 to 10, and 1 to 20, saying them aloud as they write.

(3) Read Number Words

Write down the number words one, two, three, four, and five and have children read them. When these are learned go on to the next number words.

(4) Shape Recognition

Ask children to try to find the shapes used on the screen in the room, in books, magazines. or pictures. Find the screen shapes in old magazines, cut and paste them into a scrap book of *Number Detective* shapes. Name the shapes in the scrap book.

These few ideas are meant as suggestions only, in order to assist the teacher and the parent to invent their own activities, or to adapt other activities to suit the needs of students using *Number Detective.* Both parents and teachers will probably be surprised at the variety of experimentations carried out and observations made by children using the program. During trials, for instance, one young boy was noticed repeatedly putting his finger on the screen just before Scattering the shapes. He was trying to out-guess the random Scatter procedure by trying to predict where a shape would reappear on the screen. When he 'missed', he simply tried again. These supplementary and additional activity suggestions are included in the software package, as is a large counting poster (see Appendix C) and some "Excellent" reward stickers. The whole package is presented in a sturdy three-ringed plastic binder with pockets inside both covers for easy storage of all the components. This type of package also fits readily on the school Library shelf.

IX
SOFTWARE FIELD EVALUATION

The intention of this chapter is to evaluate the *Number Detective* environment. The evaluation was conducted using four year olds in several Brisbane Pre-schools as the triallists. The method used was to observe and video-record children's behaviour in a naturalistic setting amidst their ordinary day-to-day activities. The sample sessions discussed below were selected at random and edited onto a separate video tape by an independent person. The computer corner used in each Pre-school was one of the normal activity centres available to the children in this study. The children had all previously used the computer as it was a regular part of their environment, and were familiar with its operation. Teachers were always available to assist when needed.

Ann

On Ann's first attempt to use *Number Detective* she experienced difficulty with the first screen (Figure 8·2), in that she couldn't relate the command letters on the screen with the letters on the keyboard, even though she could correctly name the letter 'A'. This difficulty is one that was foreseen by the designers, and there are several approaches to resolving it. The teacher or parent will need to decide the level of assistance to be given, if any. The possibilities include: allowing the child to keep searching the keyboard until the key is found; marking or highlighting the keys used by the program; or simply showing the child which key to press. The program design favours the first option, so that computer literacy skills are developed while using the program. This approach is favoured, provided that the

child knows that the letters on the screen correspond with the letters on the keyboard. In this case, the teacher showed Ann which key to press. The typical expected response followed, in that Ann then continued to press the **A** key until it no longer Added shapes to the line – in other words, until there were five shapes on the screen. Ann then needed to be shown the location of another key that she could try. This type of approach is to be expected until the child becomes familiar with the keyboard. Peer teaching can provide an excellent solution to this stage of computer use.

Barry

Barry Added ten shapes to the screen using the **A** key, but didn't press it exactly the correct number of times to add ten. He then Scattered them around the screen, saying, "I like scattering them." After Scattering them several times, he lined them up, exclaiming after the automatic Count procedure that, "It counts them all!" These two instances demonstrate that Barry is already correctly using some of the language of the program. He then tried the Count procedure for himself, singing up the scale with the Counting music. Barry pushed the **C** key exactly the correct number of times to count the ten shapes. He repeated this several times. This event helps to confirm the design aims for the Count procedure. The teacher sitting with Barry then indicated other commands, such as Mark, that Barry might try. The teacher showed Barry how to use the arrow keys to move to the shapes to be Marked. This was probably not necessary, given that Barry seemed to be experiencing no problems with discovering things for himself, and that the design philosophy of the program is one of non-intervention. Barry accepted the explanation quickly and set about moving to and marking three umbrellas in the line.

He then tried the **G** key, which was now available in the commands box, to see what would happen. He watched carefully as the shapes moved around the

screen, always staying in full view, and laughed delightedly when all the other shapes moved over and the umbrellas were put together at the beginning of the line. He then Marked three trains as well as the three umbrellas and found that, "They all dance! Both of them!" Barry had discovered for himself that both umbrellas and trains were Gathered because they were both still Marked. He had also attached his own label – 'dance' – to the action which takes place during the Gathering procedure. If Barry had wanted to Gather the trains alone, he would have needed to finish the previous Mark and Gather operation by pressing the **Z** key to Zap the marks under the umbrellas, before Marking and Gathering the trains. He then Marked one additional item and Gathered everything that was still Marked (seven shapes now), and finally Marked everything and Gathered again. This final sequence demonstrates the exploratory nature of the program. Barry spent a good deal of time experimenting with the Mark and Gather procedure in an attempt to discover for himself everything that it could do.

Chris and David

Chris and David were operating the program in level three and had Added twenty shapes to the screen. They had then Scattered them and were pressing non-functional keys when the teacher intervened and advised them to try one of the keys in the commands box. Chris looked at the letters available, and then tried Mark, by pressing the **M** key repeatedly until all twenty shapes were Marked. They then Gathered the shapes. During this particular Gather, which lasts about two minutes because all twenty shapes are being moved around the screen ('dancing' in Barry's terms), the teacher commented that "It's taking its time isn't it?" He was then surprised at the end of the Gather when David said, "Let's press **G** again. I like it." The teacher's comment highlights the danger of intervening and imposing adult perspectives onto what the children are doing in their exploration of the microworld environment. Chris and David then went ahead and Gathered all twenty shapes

again, despite the fact that the teacher felt that the procedure was, perhaps, too slow. This sequence illustrates the expected and desired exploratory nature of the program. The boys had not yet appreciated what the Mark procedure was capable of doing, but they tried it anyway and appreciated it at a basic level. Later they could discover, like Barry, that if you Mark only certain items, only they will 'dance' and be Gathered together at the beginning of the line.

Erica

Prior to the commencement of this session using *Number Detective*, Erica was already capable of independently turning on the computer, loading in the disk and starting up the program. She chose level three and first added twenty shapes. She had not yet mastered this feature though, as she pressed the **A** key too many times. She then Scattered the shapes twice before lining them up. She had already discovered that each Scatter did not need to be immediately followed by a Line Up command, but could be followed by another Scatter. After the automatic Count had taken place, Erica Zapped the count lines and then Marked all twenty shapes by pressing the **M** key. This indicates that she probably had not yet fully explored the possibilities of the Mark procedure and did not yet appreciate its sub-group potentialities. Erica then Zapped all the Marks from beneath each shape, and then Joined all twenty shapes into one large box, which she then Zapped. By her consistent use of the Zap procedure, Erica displayed that she fully understood its function, and displayed her intention to change from one procedure to another.

Erica next used the Take Away procedure to remove all twenty shapes from the screen. This time she pressed the **T** key exactly the right number of times to effect the procedure. She next Added one shape to the screen, Counted it a few times, Scattered it and Lined it up. She then Added a further nineteen shapes, but again did not press the **A** key exactly the correct number of times. Erica then pressed the **H** key to go to the Help screen, which she looked at briefly before

pressing the **H** key again to end Help. She had correctly applied the necessary procedure for operating the Help facility. When she had the shape screen back again, she next tried **B2Return** and boxed all twenty shapes into groups of two. She then Zapped the boxes to end the procedure.

Erica is a good example of what you would expect to find at this age, and on into Kindergarten. She was obviously computer literate in that she had no trouble finding the keys on the keyboard, and could start up the program from a 'computer off' situation. She had also progressed beyond the simple single keystroke entry function of all of the *Number Detective* procedures except Box, and clearly demonstrated that she had no problem using the higher level method of entering a command. At the same time she had obviously explored many, and perhaps all, of the functions available within the *Number Detective* microworld but had not yet attained most of the concepts available within that microworld, which is as expected. Given that she had adopted a discovery approach to using the program, it seems that she would be capable of so doing over time. It is to be noted that there was neither teacher intervention nor even presence at the computer corner during this session.

Fiona

Fiona chose level three and first Added twenty shapes. The **A** key was again not pressed the correct number of times to add exactly twenty. She next Scattered the shapes and Lined them up. The teacher suggested that she "Try **B**", but Fiona pressed the **J** key instead and Joined the first two shapes. The design philosophy does not support such intervention. It is preferable to the designers that the child first raise the question. Fiona then Zapped the Join box, demonstrating that she also knew how to end the Join procedure. The teacher persisted with her question, "What happens when you use **B**?" Fiona pressed the **B** key but as nothing apparently happened, she pressed it twice more. As the program was expecting a

number at this stage, each following **B** was received with a 'beep' from the computer. At this stage the teacher then asked, as it is expected by the designers that she should, "How many objects do you want in the box?" This is an excellent way to introduce the use of the higher level command entry method required by the Box procedure. When Fiona nominated three, she was then told to press that key, which she did. The teacher then explained that the **Return** key now needed to be pressed to actually Box the shapes. This was done and the shapes were Boxed into groups of three.

Fiona then again pressed the **B** key several times while the computer beeped. The teacher again asked how many she wanted in each box, and Fiona pressed the **3** key and then **Return**. She was now beginning to get the idea of the way this procedure worked, as she demonstrated by next pressing **B4Return**. This boxed the twenty shapes into groups of four. Fiona then stated that "We only got two in this box!", and pointed to the box at the end of the top line of shapes. She was referring to a situation similar to that shown in Figure 8·12, where the box actually wraps around to the next line. This presented a major conceptual design difficulty once more than ten shapes were available. The open-ended box was the solution adopted as it conveys the idea of the box extending to or wrapping around to the next line of shapes. When asked by the teacher why this was so Fiona, not unexpectedly, couldn't resolve the problem. The teacher then asked if there was another box which only had two shapes in it, and Fiona was able to point it out. Had she next Scattered and Lined up the Boxed shapes, she would have found that the 'split' group would have appeared elsewhere on the screen as a whole unit, as would have all the other groups. This problem obviously disrupted the flow of Fiona's thought, as she then momentarily forgot how the Box function worked, and again pressed the **B** key several times. The teacher interposed that she needed a number. Fiona then recalled the procedure and typed **5** and **Return**. The teacher again intervened at this stage by asking, "How many boxes are there?" Fiona

initially misinterpreted this question, as she counted by pointing to the number of items in each box, saying "There's five in there, and five in there, and five in there, and five in there. There's five in each box." The teacher again asked, "How many boxes are there?" and this time was told that there were four boxes. It is expected that exchanges such as this would lead children into asking such questions of themselves as they were using the program.

Gary

Gary was operating in level three, and was looking at the Join procedure for the first time. The teacher explained that the use of the arrow keys allowed for moving along the lines of shapes so that any shapes in the lines could be Joined together. Gary quickly grasped the idea and used the keys correctly to select and Join two like objects. He repeated this three times until he had four groups of two, with each group being made up of like objects. He then Scattered the groups and noticed that the Joined items were still Joined after the Scatter. He Scattered them again and noticed that they were still Joined, but had "...moved around." He next Lined them up, and found after the Line Up that there were some more like shapes sitting together now which could be Joined together. This is a result of the randomising nature of the Scatter function. He next Joined those items, ending up with six groups of two, with each group containing like objects. He then Scattered the items, again noticing that the Joined groups remained Joined, and finally Lined them up again. This sequence clearly illustrates the exploratory nature of the *Number Detective* microworld. Once Gary knew how the procedure operated, he spent quite a deal of time exploring what could be done with that procedure. This type of behaviour was seen many times during field evaluations of the software. Children would readily spend five to ten minutes simply Adding shapes to the line and then Taking them away again, for instance, before moving on to try a different procedure.

Heidi and Isabelle

Heidi and Isabelle were working in level one. Heidi Added five shapes to the screen, although she didn't press the **A** key exactly the correct number of times to do so. The teacher suggested that they try Scatter, but Heidi pressed the wrong key, so Isabelle said, "No, press the S." Heidi then Scattered the shapes and Isabelle then advised her to "Press that one", which was the **L** key, to line them up. After the Line up an automatic count always takes place, at the end of which Isabelle told Heidi to press the **Z** key (to remove the count lines), while she pointed to it. Heidi didn't try that but pressed the **J** key instead and Joined all the shapes together into one big box. "Zap!" said Isabelle (to remove the box), but Heidi pressed the **T** key instead, Taking Away one shape. By doing this she removed the Box anyway. The teacher intervened at this point, asking "What happened on the screen?", obviously expecting a comment relating to the removal of one shape and of the Joining box. Once again the response was totally unexpected from an adult's point of view. Isabelle replied that "It got longer!", referring to the commands-available box, which at this stage now had more options shown than it did before the previous Take Away. This illustrates that Isabelle was already coming to an understanding of the fact that not all commands are available at all times.

Heidi next Added one shape to the line and said, "Now I want to Scatter them. No, I want to Join them." "You have to press the J, remember", said Isabelle. Heidi then pressed the **J** key and Joined all the shapes together again, and said, "Now I want to Scatter them." The teacher again posed a question at this stage asking, "Will they Scatter all over the screen or will they stay together?" Both girls were certain that the shapes would Scatter all over the screen and were surprised when the Joined group stayed together. They Scattered again to find out what would happen the next time. Heidi then said, "Now I want to Line them up", but Isabelle wanted to Scatter them once more. Having seen the result (the Joined group was still Joined), she then agreed to Line them up. The question posed by

the teacher here was probably unnecessary as there was every likelihood that the girls would have noticed the result of the Scatter for themselves anyway, if not this particular time, then perhaps later. The teacher certainly took away from the girls the chance to discover a new 'rule' for themselves by hinting that he already knew the answer to his own question. It is the intention of the design of the program that these new 'rules' are discovered by the children for themselves, rather than that they are led towards a discovery or, worse, that they are told what will be the result of a particular procedure.

Summary

When left to their own devices, it seems that there is little doubt that children as young as four years of age can cope admirably, both with the normal operation of a microcomputer and with the functions available in the *Number Detective* microworld. The cases examined above illustrate that the keyboard skills needed to operate the program are readily learned by young children, including the higher level skill involving the use of the **return** key. What is also illustrated in several cases is that there is really no need for teachers to intervene in most instances, the one exception perhaps being when a totally different procedure, such as the use of the **return** key, is being approached for the first time.

The range and permanence of concepts learned by children using the *Number Detective* microworld are not to be investigated by this study, although a companion study of such issues is in the early stages of planning. It should be remembered that this section of the present study sets out only to explore the efficacy of the *Number Detective* microworld as a learning environment.

X
CONCLUSION

The availability of new computer technology opens up the possibility, for the first time since the introduction of the text book during the 18th century, of implementing radical changes in teaching methods. As is attested by the recent (August, 1988) reduction of teacher numbers by the Government of New South Wales, "... in many countries we are moving, whether we like it or not, towards systems with fewer teachers. The way we can overcome this problem and achieve [high-quality education] is by the use of technology." (Bork. In Towie, 1988, 33) Bork asserts that the increasing costs associated with education would make it necessary for many countries to examine new teaching methods which introduce economies of scale by allowing for the restructuring of educational systems. Bork believes that traditional teaching methods should be replaced with computer-based learning programs which are better able to meet the needs of individual students. He further claims that "... the existing teaching system is inhumane: it can't deal with the vast range of differences in individual students." (Bork. In Towie, 1988, 33)

The first major hurdle to overcome in the implementation of such a plan is that of making available to students the computer power with which to be able to take advantage of such a computer-based system. The rationale of the New South Wales Government in reducing teacher numbers is that of cost reduction, given that the average teacher wage is currently in the vicinity of $30,000 per annum. By reducing teacher numbers by two thousand, as planned, the Government will save approximately $60 million each year. The savings made in one year alone could

provide more than 40,000 computers for individual student use at a cost of just over $115 per year per student, averaged out over the thirteen years of primary and secondary schooling. If a further annual allowance was provided on a per capita basis for the purchase of software and for hardware maintenance, upgrading and replacement if necessary, the total amount would still only represent a very small percentage of total public expenditure on education. Papert (1980, 17) argues that the use of computers in education could also make other aspects of education cheaper:

> Schools might be able to reduce their cycle from thirteen to twelve years; they might be able to take advantage of the greater autonomy the computer gives students and increase the size of classes by one or two students without decreasing the personal attention each student is given. Either of these two moves would "recuperate" the computer cost.

It is ironic that the reduction of teacher numbers in New South Wales will also result in an increase of class sizes by one or two students.

The second major hurdle to overcome involves the education of trainee teachers, and the re-education of the present teacher workforce. Teacher training methods would need to be re-assessed to meet the changed role of the teacher in an environment which contains more computers (hopefully, up to one per student), but which at the same time will probably contain more students in each class. Trainee teachers would then need to be aware of the methods available to make best use of such an environment, with the aim always being to provide an optimum learning environment for all students. Bork (In Towie, 1988) would go further and move schools away from structured grades and classes toward an open style of learning based on students mastering subjects at their own pace, while making curricula more relevant to the needs of society. Teachers already in the workforce need assistance in focussing on just what are the most productive and valuable educational uses of computers in schools. In the "Considerations for Schools"

section of its *General Policy Statement*, the New South Wales Department of Education offers only broad guidelines which offer little support for the teacher:

> Schools are encouraged to explore computer techniques which will add new dimensions to the student's experience. Planning must, as always, begin with a consideration of the educational aims which it is hoped to achieve through the use of computers.
> In developing policies on the use of computers for learning, schools should be aware that computer systems may assist students to obtain better understanding of some significant concepts in various curriculum areas. Such experiences also contribute indirectly to increasing student awareness of the wider uses of computers in society.
> The wide range of uses of computers in learning needs to be explored so that the implications can be fully understood and decisions made. (1983, 9)

At the time that this statement was issued, the majority of teachers in the teaching system had never seen a computer, let alone used one to the extent of being able to 'explore computer techniques' and construct a meaningful curriculum around one. Indeed, many schools found that there was no teacher on the staff who even knew how to unpack and assemble the computer when it arrived on the doorstep. In most schools, unfortunately, the situation has changed little even today.

The intent of this study has been to attempt to overcome some of these difficulties by examining the questions faced by the teacher when confronted by this new technology. The chapter on Computer Assisted Learning provides the teacher with an understanding of the types of uses to which a computer can be put in an educational environment. By an examination of these different uses, the teacher should be better able to decide what methods are best suited to various curriculum areas and topics. This decision should also be made easier after looking at the strengths and weaknesses of the various software types. The chapter on Theories of Learning and Software Design provides the teacher with an understanding of the way in which CAL software has been developed from a learning theory point of view. It also provides models for the development of three different types of software so that the teacher can gain an appreciation of the design approach to be taken in the development of such software. This is an important factor, as there

needs to be a far greater teacher input into the development of such software than is currently the case.

The chapter on Theories of Intellectual Development provides the teacher with a broad overview of the developmental stages of human knowledge, as they are currently described. The intent of providing such an overview is to assist the teacher to match the performance and needs of each student with the software that is available. Such an overview also provides the teacher with information which may assist in structuring general classroom lessons away from the computer. The two chapters on Software Evaluation provide the teacher with information on evaluating computer software, in order to be able to determine whether or not a particular piece of software is suitable for use for the requirements of the students in a given curriculum area. The first of these chapters looks broadly at the factors that should be considered in making such an evaluation, while the second is comprised of two sample evaluations.

As stated previously, there is an urgent need for teachers to become more heavily involved in the software design process, particularly as software becomes more and more educationally sophisticated. In the early days of software design, the general quality of educational software was poor. This was because it was either designed by a computer 'buff' or even a computer scientist who had little or no knowledge of the child's educational requirements, or by a teacher who had sound educational ideas, but who had little or no computer experience. The first type tended to be very good technically, but unsound educationally, while the latter tended to be educationally sound, but technically very poor. The middle ground design team consists of teachers as the educational designers, so that the program is educationally sound, and of computer scientists as the programmers, so that the program is technically sound.

The chapters on Software Design and Field Evaluation are intended to provide the teacher with a deeper insight into the demands of the software design problem so that it can be more fully understood as an educational design problem and so that, should anyone wish, ideas can be picked up and further developed in relation to the creation of new computer software. The educational computer environment needs to be composed in such a way as to enable children to explore knowledge naturally. In order to be able to do this, appropriate materials need to be available. In order to create these materials, expertise is needed. Papert (1980) breaks this expertise down into three categories: knowledge about computers, knowledge about subject domains, and knowledge about people. Papert is of the opinion that the essential remaining problem in regard to the future of computers and education is the problem of the supply of people who will develop this material. The genesis of this study was an attempt to provide the teaching community with a reference which may in some way contribute toward a solution to that essential remaining problem.

BIBLIOGRAPHY

Adams, T. 1986. Towards a theory of microworlds. In Salvas, A. D. and Dowling, C. (Eds). 1986.

Anderson, J. R. 1985. *Cognitive Psychology and Its Implications*. New York: W. H. Freeman and Company.

Anglin, J. M. (Ed.). 1974. *Beyond the Information Given*. London: George Allen & Unwin.

Angus, J. 1985. The Luria model of information processing. *Australian Journal of Educational Technology*. Volume 1, Number 1, January.

Appel, L. F., Cooper, R. G. and McCarrell, N., et al. 1972. The development of the distinction between perceiving and memorizing. *Child Development*. Volume 43, Number 4.

Arundell, C. and Coote, T. 1986. Computers and Home Economics. In Frederick, B. 1986.

Bandura, A. 1971. *Social Learning Theory*. Morristown, New Jersey: General Learning Press.

Bandura, A. 1977. *Social Learning Theory*. Englewood Cliffs, New Jersey: Prentice-Hall Inc.

Billings, K. 1983. Developing mathematical concepts with microcomputer activities. *Arithmetic Teacher*. Volume 30, Number 6, February.

Boden, M. 1979. *Piaget*. Brighton: The Harvester Press.

Boden, M. A. 1980. The social implications of intelligent machines. In Forester, T. (Ed.), 1980.

Boulding, K. E. 1961. National image and international systems. *Journal of Conflict Resolution*. Volume 3.

Bower, G. H. and Hilgard E. R. 1981. *Theories of Learning*. Englewood Cliffs, New Jersey: Prentice-Hall.

Bowersock, D. C. 1983. *Education Development Center, Inc. 25th Year Report*. Newton: Education Development Center.

Bresnan, J., Miller, G. and Halle, M. (Eds). 1978. *Linguistic Theory and Psychological Reality*. Cambridge, Mass.: MIT Press.

Broadbent, D. E. 1961. *Perception and Communication*. London: The Scientific Book Guild.

Brown, J. S. and Burton, R. R. 1978. Diagnostic models for procedural bugs in basic mathematical skills. *Cognitive Science*. Volume 2, 1978.

Brown, M. A. 1978. Cognitive development and the learning of mathematics. In Floyd, A. (Ed.). 1979.

Bruner, J. S. 1971. *Toward a Theory of Instruction.* Cambridge: Harvard University Press.

Bruner, J. S. 1974a. Readiness for learning. In Anglin, J. M. (Ed.). 1974.

Bruner, J. S. 1974b. The act of discovery. In Anglin, J. M. (Ed.). 1974.

Bruner, J. S. 1974c. The course of cognitive growth. In Anglin, J. M. (Ed.). 1974.

Bruner, J. S. 1974d. The growth of mind. In Anglin, J. M. (Ed.). 1974.

Bruner, J. S. 1974e. The process of concept attainment. In Anglin, J. M. (Ed.). 1974.

Bruner, J. S., Goodnow, J. J. and Austin, G. A. 1967. *A Study of Thinking.* New York: John Wiley & Sons.

Burton, R. R. 1982. Diagnosing bugs in a simple procedural skill. In Sleeman, D. and Brown, J. S. (Eds), 1982.

Burton, R. R. and Brown, J. S. 1982. An investigation of computer coaching for informal learning activities. In Sleeman, D. and Brown, J. S. (Eds), 1982.

Carey, S. 1974. Cognitive competence. In Floyd, A. (Ed.). 1979.

Carey, S. 1978. The child as a word learner. In Bresnan, J., Miller, G. and Halle, M. (Eds). 1978.

Carey, S. 1984. Cognitive development: The descriptive problem. In Gazzaniga, M. S. (Ed.). 1984

Case, R. 1978. Piaget and beyond: Towards a developmentally based theory and technology of instruction. In Glaser, R. (Ed.). 1978.

Chambers, J. A. and Sprecher, J. W. 1980. Computer assisted instruction: Current trends and critical issues. *Communications of the ACM.* Volume 23, Number 6.

Chambers, J. A. and Sprecher, J. W. 1983. *Computer Assisted Instruction: Its Use in the Classroom.* Englewood Cliffs, New Jersey: Prentice-Hall.

Chambers, S. M. 1987. Cognitive processes used by children writing Logo programs. *British Journal of Developmental Psychology.* Volume 5, Number 2.

Chen, M. and Paisley, W. (Eds). 1985. *Children and Microcomputers.* London: SAGE Publications.

Chi, M. T. H., Glaser, R. and Rees, E. 1982. Expertise in problem solving. In Sternberg, R. J. (Ed.). 1982

Cole, M., Frankel, F., and Sharp, D. 1971. Development of free recall learning in children. *Developmental Psychology.* Volume 4.

Computers in Schools: A General Policy Statement. 1983. Sydney: New South Wales Department of Education.

Cormier, S. M. 1986. *Basic Processes of Learning, Cognition, and Motivation.* Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Cratty, B. J. 1968. *Movement Behaviour and Motor Learning.* Philadelphia: Lea & Febiger.

Cruickshank, W. M. and Hallahan, D. P. (Eds). 1975. *Perceptual and Learning Disabilities in Children.* (Volume 2). Syracuse, New York: Syracuse University Press.

Curriculum for Primary School Mathematics. 1972. Sydney: New South Wales Department of Education.

Dale, P. S. 1976. *Language Development Structure and Function.* New York: Holt, Rinehart & Winston.

Dailhou, P. 1986a. Using adventure games. In Frederick, B. 1986.

Dailhou, P. 1986b. Computers in the writing classroom. In Frederick, B. 1986.

DeCecco, J. P. (Ed.). 1964. *Educational Technology.* New York: Holt, Rinehart and Winston.

Derricott, R. and Blyth, A. 1978. Cognitive development: The social dimension. In Floyd, A. (Ed.). 1979.

Dickson, D. 1974. *Alternative Technology and the Politics of Technical Change.* Glasgow: Fontana/Collins.

doing something you've never done before. 1987. Sydney: Computer Education Group of New South Wales.

Downs, R. M. and Stea, D. (Eds). 1973. *Image and Environment.* London: Edward Arnold.

Durrett, J. and Trezona, J. 1982. How to use color displays effectively. *BYTE.* Volume 7, Number 4, April.

Elio, R. E. and Reutener, D. B. 1978. Color context as a factor in encoding and as an organization device for retrieval of word lists. *The Journal of General Psychology.* Volume 99.

Farley, F. H. and Grant, A. P. Arousal and cognition: Memory for color versus black and white multimedia presentation. *The Journal of Psychology.* Volume 94, September.

Farnham-Diggory, S. (Ed.). 1972a. *Information Processing in Children.* New York: Academic Press.

Farnham-Diggory, S. 1972b. The development of equivalence systems. In Farnham-Diggory, S. (Ed.). 1972.

Flavell, J. H. 1977. *Cognitive Development.* Englewood Cliffs, New Jersey: Prentice-Hall.

Flavell, J. H., Friedrichs, A. G. and Hoyt, J. D. 1970. Developmental changes in memorization processes. *Cognitive Psychology*. Volume 1, Number 4.

Flavell, J. H. and Wellman, H. M. 1977. Metamemory. In Kail, R. V. and Hagen, J. W. (Eds). 1977.

Floyd, A. (Ed.). 1979. *Cognitive Development in the School Years*. London: Croom Helm

Forester, T. (Ed.). 1980. *The Microelectronics Revolution*. Oxford: Basil Blackwell.

Forwood, C. J. 1986, LOGO in Kindergarten. In Frederick, B. 1986.

Franks, A. 1984. In the year 2000, school will be for play only. *The Australian*. July 24, 1984.

Frederick, B. 1986. *What To Do With What You've Got*. Sydney: Computer Education Group of NSW.

Frost, G. 1986. LOGO activities for Kindergarten. In Frederick, B. 1986.

Fuson, K. C. and Hall, J. W. 1983. The acquisition of early number word meanings. In Ginsburg, H. P. (Ed.). 1983.

Gage, N. L. (Ed.). 1967. *Handbook of Research on Teaching*. Chicago: Rand McNally and Company.

Gagné, R. M. and Briggs, L. J. 1979. *Principles of Instructional Design*. New York: Holt, Rinehart and Winston.

Gare, R. 1982a. *Educational Software Development*. Brisbane: Queensland Department of Education.

Gare, R. (Ed.). 1982b. *Computer Download*. Brisbane: Queensland Department of Education. July.

Gare, R. (Ed.). 1982c. *Computer Download*. Brisbane: Queensland Department of Education. November.

Gare, R. (Ed.). 1983. *MECC Computer Based Learning Materials*. Brisbane: Queensland Department of Education.

Gazzaniga, M. S. (Ed.). 1984 *Handbook of Cognitive Neuroscience*. New York: Plenum Press.

Gholson, B., Levine, M. and Phillips, S. 1972. Hypothesis strategies and stereotypes in discrimination learning. *Journal of Experimental Child Psychology*. Volume 13.

Gholson, B. and McConville, K. 1974. Effects of stimulus differentiation training upon hypotheses, strategies, and stereotypes in discrimination learning among kindergarten children. *Journal of Experimental Child Psychology*. Volume 18.

Gilding, A. and Pearce, J. M. LOGO and the exploration of some concepts in science. In Salvas, A. D. and Dowling, C. (Eds). 1986.

Ginsburg, H. P. (Ed.). 1983. *The Development of Mathematical Thinking.* New York: Academic Press.

Glaser, R. (Ed.). 1978. *Advances in Instructional Psychology.* (Volume 1). Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Goodnow, J. 1972. Rules and repertoires, rituals and tricks of the trade: social and informational aspects to cognitive and representational development. In Farnham-Diggory, S. (Ed.). 1972.

Greig, I. B. and Stace, R. J. 1985a. *Addition Made Easy.* Wollongong: Spinifex Software.

Greig, I. B. and Stace, R. J. 1985b. *Subtraction Made Easy.* Wollongong: Spinifex Software.

Greig, I. B. and Stace, R. J. 1985c. *Long Multiplication Made Easy.* Wollongong: Spinifex Software.

Greig, I. B. and Stace, R. J. 1985d. *Long Division Made Easy.* Wollongong: Spinifex Software.

Greig, I. B. and Stace, R. J. 1985e. *Number Detective.* Wollongong: Spinifex Software.

Greig, I. B. and Stace, R. J. 1985f. *Number Explorer.* Wollongong: Spinifex Software.

Hagen, J. W. 1967. The effects of distraction on selective attention. *Child Development.* Volume 38, Number 3.

Hagen, J. W. 1972. Strategies for remembering. In Farnham-Diggory, S. (Ed.). 1972.

Hagen, J. W. and Kail, R. V. 1973. Facilitation and distraction in short-term memory. *Child Development.* Volume 44.

Hagen, J. W. and Kail, R. V. 1975. The role of attention in perceptual and cognitive development. In Cruickshank, W. M. and Hallahan, D. P. (Eds). 1975.

Hagen, J. W., Jongeward, R. H., and Kail, R. V. 1975. Cognitive perspectives on the development of memory. In Floyd, A. (Ed.). 1979.

Hagen, J. W. and Stanovich, K. E. 1977. Memory: Strategies and acquisition. In Kail, R. V. and Hagen, J. W. (Eds). 1977.

Hall, N. and Brown, H. 1986. Process writing, computers and a confident teacher. In Salvas, A. D. and Dowling, C. (Eds). 1986.

Hallahan, D. P., Kauffman, J. M., and Ball, D. W. 1973. Selective attention and cognitive tempo of low achieving and high achieving sixth grade males. *Perceptual and Motor Skills.* Volume 36.

Handle With Care: A Guide to identifying and eliminating bias in software. 1985. Sydney: New South Wales Department of Education.

Hardwick, D. A., McIntyre, C. W., and Pick, H. L. 1976. The content and manipulation of cognitive maps in children and adults. *Monographs of the Society for Research in Child Development.* Volume 4, Number 3, Serial Number 166.

Harmon, P. 1985. Instructional software: A basic taxonomy. *Performance & Instruction Journal.* Volume 24, Number 5, June.

Haynes, F. 1986. Further comments: of computers, cabbage patch dolls, and creativity. *The Institute of Art Education Journal.* Volume 10, Number 2, August.

Hergenhahn, B. R. 1976. *An Introduction to Theories of Learning.* Englewood Cliffs, New Jersey: Prentice-Hall.

Herman, J. F. and Siegel, A. W. 1978. The development of cognitive mapping of the large-scale environment. *Journal of Experimental Child Psychology.* Volume 26, Number 3.

Hetherington, E. M. (Ed.). 1975. *Review of Child Development Research.* (Volume 5). Chicago: University of Chicago Press.

Hill, S. & Johnston, R. (Eds). 1983. *Future Tense?* St. Lucia: University of Queensland Press.

Hoelscher, K. 1986. Teaching and learning with the computer as a problem solving tool. In Frederick, B. 1986.

Hofmeister, A. and Maggs, A. 1984. *Microcomputer Applications in Education and Training.* Sydney: Holt, Rinehart and Winston.

Honig, W. K. (Ed.). 1966. *Operant Behaviour: Areas of Research and Application.* New York: Appleton-Century-Crofts.

Horn. J. 1987. A context for understanding information processing studies of human abilities. In Vernon, P. A. (Ed.). 1987.

Hunt, E. 1986. The information processing approach to intelligence. In Newstead, S. E., Irvine, S. H. and Dann, P. L. (Eds). 1986.

Hurlock, E. B. 1964. *Child Development.* New York: McGraw-Hill Book Company.

Huttenlocher, J. and Presson, C. C. 1973. Mental rotation and the perspective problem. *Cognitive Psychology.* Volume 4.

Isaac, S. and Michael, W. B. (Eds). 1984. *Handbook in Research and Evaluation.* San Diego: EdITS Publishers.

James, L. 1986. LOGO study. In Salvas, A. D. and Dowling, C. (Eds). 1986.

Jonassen, D. H. 1985. Interactive lesson design: A taxonomy. *Educational Technology.* Volume 25, Number 6, June.

Jones, B. 1982. *Sleepers, Wake!* Melbourne: Oxford University Press.

Kail. R. V. and Hagen, J. W. (Eds). 1977. *Perspectives on the Development of Memory and Cognition.* Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Kail, R. V. and Siegel, A. W. 1977. The development of mnemonic encoding in children: From perception to abstraction. In Kail, R. V. and Siegel, A. W. 1977.

Kantowski, M. G. 1983. The microcomputer and problem solving. *Arithmetic Teacher.* Volume 30, Number 6, February.

Kaplan, S. 1973. Cognitive maps in perception and thought. In Downs, R. M. and Stea, D. (Eds). 1973.

Keating, D. P. and MacLean, D. J. 1987. Cognitive processing, cognitive ability, and development: A reconsideration. In Vernon, P. A. (Ed.). 1987.

Keller, A. 1987. *When Machines Teach.* New York: Harper & Row.

King, R. C. 1985. *For Whom the Bell Rings: The Reality of Schooling.* Proceedings of the Australian College of Education (Townsville Regional Group), Townsville, Australia.

Klahr, D. and Wallace, J. G. 1976. *Cognitive Development: An Information - processing view.* Hillsdale, New Jersey: Lawrence Erlbaum Associates

Kobasigawa, A. 1977. Retrieval strategies in the development of memory. In Kail, R. V. and Siegel, A. W. 1977.

Lamoureux, K. 1986. Meeting the needs of girls in computer education. In Frederick, B. 1986.

Lewis, M. 1975. The development of attention and perception in the infant and young child. In Cruickshank, W. M. and Hallahan, D. P. (Eds). 1975.

Liben, L. S. 1977. Memory in the context of cognitive development: The Piagetian approach. In Kail, R. V. and Siegel, A. W. 1977.

Lovell, K. 1970. *An Introduction to Human Development.* London: Macmillan.

Matson, M. 1985. *Dragon World.* Barnstaple, Devon: 4MATION.

Marsland, A. 1986. Creating and using databases in geography classes. In Salvas, A. D. and Dowling, C. (Eds). 1986.

McDougall, A. and Squires, D. 1986. Student control in computer-based learning environments. In Salvas, A. D. and Dowling, C. (Eds). 1986.

Mevarech, Z. R. and Ben-Artzi, S. 1987. Effects of CAI with fixed and adaptive feedback on children's mathematics anxiety and achievement. *The Journal of Experimental Education.* Volume 56, Number 1.

Miller, G. 1977. *Spontaneous Apprentices: Children and Language.* New York: Seabury Press.

Moely, B. E. 1977. Organizational factors in the development of memory. In Kail, R. V. and Siegel, A. W. 1977.

Molloy, S. 1986. Adventuring with the Robinson's. In Frederick, B. 1986.

Mulock, R. 1983. A message from the Minister. *Computers in Schools: A General Policy Statement.* Sydney: New South Wales Department of Education.

Myers, N. A. and Perlmutter, M. 1978. Memory in the years from two to five. In Ornstein, P. A. (Ed.). 1978.

Nesbitt-Hawes, P. J. and Nesbitt-Hawes, S. D. 1986. More than just discovering dinosaurs. In Frederick, B. 1986.

Neisser, U. 1976. *Cognition & Reality.* San Francisco: W. H. Freeman and Company.

Newstead, S. E., Irvine, S. H. and Dann, P. L. (Eds). 1986. *Human Assessment: Cognition and Motivation.* Lancaster: Martinus Nijhoff Publishers.

Norman. D. A. 1976. *Memory and Attention.* New York: John Wiley & Sons.

Oakley, J. 1986a. Can LOGO live up to its promise? In Frederick, B. 1986.

Oakley, J. 1986b. Is there more to LOGO than Piaget and powerful ideas? In Salvas, A. D. and Dowling, C. (Eds). 1986.

O'Brien, L. and Kerr, D. 1986. The data base package: Fact not fiction. In Frederick, B. 1986.

Olds, H. F. 1986. Critical thinking and problem solving in the language arts. In Frederick, B. 1986.

Ornstein, P. A. (Ed.). 1978. *Memory Development in Children.* New York: John Wiley & Sons.

Ornstein, P. A. and Naus, M. J. 1985. Effects of the knowledge base on children's memory strategies. In Reese, H. 1985.

Osser, H. and Rudmin, F. 1986. Metacognition, social context and personal factors in arithmetic problem solving. In Newstead, S. E., Irvine, S. H. and Dann, P. L. (Eds). 1986.

Papert, S. 1980. *Mindstorms - Children, Computers and Powerful Ideas.* Brighton: Harvester Press.

Paris, S. G. and Lindauer, B. K. 1977. Constructive aspects of children's comprehension and memory. In Kail, R. V. and Siegel, A. W. 1977.

Parrill-Burnstein, M. 1981. *Problem Solving and Learning Disabilities: An Information Processing Approach.* New York: Grune and Stratton.

Peterson, P. L. and Swing, S. R. 1982. Beyond time on task: Students' reports of their thought processes during classroom instruction. *The Elementary School Journal.* Volume 82, Number 5, May.

Piaget, J. 1960. *The Child's Conception of the World.* London: Roultedge & Kegan Paul.

Piaget, J. and Inhelder, B. 1969. *The Psychology of the Child.* New York: Basic Books

Pick, H. L. and Saltzman, E. 1978. *Modes of Perceiving and Processing Information.* Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Pirie, I. G. and Jecks, D. 1970. *Continuous Progress in Mathematics for New South Wales Schools: Kindergarten Book.* Perth: Carroll's Pty Ltd.

Pirie, I. G. and Jecks, D. 1971. *Continuous Progress in Mathematics for New South Wales Schools: Book One.* Perth: Carroll's Pty Ltd.

Plato Computer Based Education. 1983. *Microcomputer Author's Guide.* San Diego: Control Data Publishing.

Pula, F. J. and Goff, R. J. (Eds). 1972. *Technology in Education: Challenge and Change.* Belmont, California: Wadsworth Publishing Company.

Quinn, M. 1986. The use of spreadsheets in primary mathematics. In Salvas, A. D. and Dowling, C. (Eds). 1986.

Reese, H. (Ed.). 1975. *Advances in Child Development and Behaviour.* (Volume 10). New York: Academic Press.

Reese, H. (Ed.). 1985. *Advances in Child Development and Behaviour.* (Volume 19). New York: Academic Press.

Reinecke, I. 1982. *Micro Invaders.* Ringwood: Penguin Books Australia Ltd.

Resnick, L. B. 1983. A developmental theory of number understanding. In Ginsburg, H. P. (Ed.). 1983.

Rieber, M. 1969. Hypothesis testing in children as a function of age. *Developmental Psychology.* Volume 1, Number 4.

Robinson, S. (Ed.). 1983a. *Guide to the Software Assessment Procedure, Reviewer Document #1: Courseware.* Bethesda: NEA Educational Computer Service.

Robinson, S. (Ed.). 1983b. *Guide to the Software Assessment Procedure, Reviewer Document #2: Applications Software.* Bethesda: NEA Educational Computer Service.

Rohwer, W. D. and Dempster, F. N. 1977. Memory development and educational processes. In Kail, R. V. and Siegel, A. W. 1977.

Rushby, N. J. 1979. *An Introduction to Educational Computing.* London: Croom Helm.

Sahakian, W. S. 1976. *Learning: Systems, Models and Theories.* Chicago: Rand McNally College Publishing.

Sale, A. 1982. Information technology and education. *Technological Change: Impact of Information Technology 1982*. Canberra: Australian Government Publishing Service.

Salvas, A. D. and Dowling, C. (Eds). 1986. *Computers in Education: On the Crest of a Wave?*. Balaclava: Computer Education Group of Victoria.

Saunders, R. and Whiddon, G. 1986. Data bases - Where should they be going? In Frederick, B. 1986.

School keyboard blamed for girl's disenchantment. *The Australian*. June 7, 1988.

Shantz, C. U. 1975. The development of social cognition. In Hetherington, E. M. (Ed.). 1975.

Skinner, B. F. 1958. Teaching machines. *Science,* Vol. 128, No. 3330.

Skinner, B. F. 1966. Operant behaviour. In Honig, W. K. (Ed.). 1966.

Skinner, B. F. 1968. *The Technology of Teaching*. New York: Appleton-Century-Crofts.

Sleeman, D. and Brown, J. S. (Eds), 1982. *Intelligent Tutoring Systems*. New York: Academic Press.

Smith, P. 1986. The gears of childhood: A constructive approach to problem solving and computers in education. In Salvas, A. D. and Dowling, C. (Eds). 1986.

Sternberg, R. J. (Ed.). 1982a. *Advances in the Psychology of Human Intelligence*. (Volume 1) Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Sternberg, R. J. 1982b. A componential approach to intellectual development. In Sternberg, R. J. (Ed.). 1982.

Sternberg, R. J. 1986. A triarchic theory of human intelligence. In Newstead, S. E., Irvine, S. H. and Dann, P. L. (Eds). 1986.

Stevenson, J. C. 1986. Adaptability: Theoretical considerations. *Journal of Structural Learning*. Volume 9.

Stonier, T. 1984. The Revolution in Education. *QUICK - Journal of the Computer Education Group of Queensland.* Number 13, March, 1984.

Tatnall, A. and Davey, B. 1986. Conceptual development through robotics: Information technology in practice. In Salvas, A. D. and Dowling, C. (Eds). 1986.

Teaching, Learning and Computers. 1983. Canberra: Commonwealth Schools Commission.

Teaching, Learning and Computers: 1984 Information Kit. 1984. Canberra: Australian Government Publishing Service.

Technological Change: Impact of Information Technology 1982. Canberra: Australian Government Publishing Service.

Thompson, C. L. 1983. The microcomputer - new forum, old questions. *Education Development Center, Inc. 25th Year Report.* Newton: Education Development Center.

Tighe, T. J. 1982. *Modern Learning Theory: Foundations and Fundamental Issues.* New York: Oxford University Press.

Tobin, K. 1987. The role of wait time in higher cognitive level learning. *Review of Educational Research* (US). Volume 57, Number 1, Spring.

Toffler, A. 1970. *Future Shock.* London: Pan Books Ltd.

Towie, M. 1988. Educator urges more effective way of teaching. *The Australian.* July 26, 1988.

Vernon, P. A. (Ed.). 1987a. *Speed of Information-Processing and Intelligence.* Norwood, New Jersey: Ablex Publishing Corporation.

Vernon, P. A. 1987b. New developments in reaction time research. In Vernon, P. A. (Ed.). 1987a.

Verschaffel, L. 1986. New tendencies in western research on learning and instruction. *Education and Society.* Volume 3, Number 2.

Vogler, L. 1983. Computers in the classroom - Secondary school case studies. *Guidelines.* Brisbane: Queensland Department of Education.

Wellington, J. J. 1985. *Children, Computers and the Curriculum.* London: Harper & Row.

Wells, G. (Ed.). 1985. *Language Development in the Pre-school Years.* Cambridge: Cambridge University Press.

White, M. A. (Ed.). 1983. *The Future of Electronic Learning.* London: Lawrence Erlbaum Associates.

Williams, R. 1974. *Television - Technology and Cultural Form.* Glasgow: Fontana/Collins.

Winne, P. H. 1985. Steps toward promoting cognitive achievements. *The Elementary School Journal.* Volume 85, Number 5, May.

Winne, P. H. and Marx, R. W. 1982. Students' and teachers' views of thinking processes for classroom learning. *The Elementary School Journal.* Volume 82, Number 5, May.

Wright, J. C. and Vliestra, A. G. 1975. The development of selective attention: From perceptual exploration to logical search. In Reese, H. (Ed.). 1975.

APPENDIX A

SAMPLE SOFTWARE EVALUATION FORM: Part A

Purchasing Information

Program Title: _____

Publisher : _____

Address: _____

_____

Phone: _____

Evaluation copy supplied by: _____

Related programs: _____

_____

_____

Available for:	Apple][+	[ ]	Apple][e	[ ]
	Apple][c	[ ]	Commodore 64	[ ]
	Microbee	[ ]	BBC	[ ]
	IBM PC Jr	[ ]	Macintosh	[ ]
	Others: _____			
Availability	License agreement			[ ]
	Purchase			[ ]
	Public domain			[ ]
	School kits			[ ]
	Networking			[ ]
	Others: _____			
Pricing:	School license fee			$
	RRP			$
	Other:			

Back up policy: Free [ ]

Small fee ($) [ ]

Exchange [ ]

Other: _____

Update policy: Free [ ]

Small fee ($) [ ]

Exchange [ ]

Other: _____

Program version tested (Eg: V1.0): _____

Program evaluated by: _____

Occupation: _____

Expertise in Education/Computers in Education: _____

_____

_____

_____

Date program evaluated: _____

SAMPLE SOFTWARE EVALUATION FORM: Part B

Program Package Description

Packaging description: _____

_____

Support materials provided:

User Guide	[ ]
Reference Manual	[ ]
Student Guide	[ ]
Additional Activities	[ ]
Other: _____	

Hardware required for this package:

Computer: _____

Minimum memory: _____

	Essential	Optional
Printer	[ ]	[ ]
Single disk drive	[ ]	[ ]
Dual disk drives	[ ]	[ ]
Cassette player	[ ]	[ ]
Colour monitor	[ ]	[ ]

Other: _____

SAMPLE SOFTWARE EVALUATION FORM: Part C

Program Description

Program type:

Adventure Game	[ ]
Data Base	[ ]
Demonstration	[ ]
Diagnostic	[ ]
Drill and practice	[ ]
Educational Game	[ ]
MicroWorld	[ ]
Simulation	[ ]
Tutorial	[ ]
Word processor	[ ]
Other: _____	

Suitable for use by:

Individual	[ ]
Group	[ ]
Class	[ ]
Other: _____	

Suitable for use for:

Concept Introduction	[ ]
Concept Development	[ ]
Concept Review	[ ]
Skill Introduction	[ ]
Skill Development	[ ]
Skill Review	[ ]
Other: _____	

Target Audience:

Age: _____

Year/Class: _____

Special audience: _____

Curriculum areas treated: _____

_____

Topics treated: _____

_____

_____

Brief description of how those topics are treated:

_____

_____

_____

_____

_____

SAMPLE SOFTWARE EVALUATION FORM: Part D

Evaluation Checklist

Content

Objectives fully and clearly stated	[Y N NA]
Objectives worthwhile	[Y N NA]
Objectives relevant to the curriculum	[Y N NA]
Target audience clearly defined	[Y N NA]
Vocabulary level appropriate to audience	[Y N NA]
Content accurate and appropriate	[Y N NA]
Teaching method appropriate	[Y N NA]
Content free from bias	[Y N NA]

Comments: _____

_____

_____

_____

Program features

Program error trapped and easy to use	[Y N NA]
Presentation clear and easy to understand	[Y N NA]
Feedback is appropriate and not a distraction	[Y N NA]
Rate of progress to suit individual	[Y N NA]
Program can be used independently	[Y N NA]

Comments: _____

_____

_____

_____

Special features

"HELP" facility is available [Y N NA]

Print out facilities available [Y N NA]

Program editor available [Y N NA]

Comments: _____

_____

_____

_____

Classroom application

Preparation required by teacher [Y N NA]

Preparation required by student [Y N NA]

Convenient to use in the classroom [Y N NA]

Results provided for the teacher and student [Y N NA]

Results kept by the program [Y N NA]

Follow-up required by the teacher [Y N NA]

Follow-up required by the student [Y N NA]

Comments: _____

_____

_____

_____

Support materials

User Guide instructions adequate and accurate [Y N NA]

Teaching ideas provided [Y N NA]

Additional activities appropriate and effective [Y N NA]

Educational design accurate [Y N NA]

Comments: _____

_____

_____

_____

SOFTWARE EVALUATION FORM: Part E

Evaluation Summary

Content: Rating [5 4 3 2 1]

Comments: _____

_____

_____

_____

Program features: Rating [5 4 3 2 1]

Comments: _____

_____

_____

_____

Special features: Rating [5 4 3 2 1]

Comments: _____

_____

_____

_____

Classroom application: Rating [5 4 3 2 1]

Comments: _____

_____

_____

_____

Support materials: Rating [5 4 3 2 1]

Comments: _____

_____

_____

_____

Overall impression: Total [____]

Comments: _____

_____

_____

_____

APPENDIX B

Bug Frequency Table (Brown and Burton, 1978, 182)

The 14 most frequently occurring bugs in a group of 1325 students

57 students used: BORROW/FROM/ZERO (103 - 45 = 158)

When borrowing from a column whose top digit is 0, the student writes 9, but does not continue borrowing from the column to the left of the 0.

54 students used: SMALLER/FROM/LARGER (253 - 118 = 145)

The student subtracts the smaller digit in a column from the larger digit regardless of which one is on top.

50 students used: BORROW/FROM/ZERO and LEFT/TEN/OK (803-508 =395)

The student changes 0 to 9 without further borrowing unless the 0 is part of a 10 in the left part of the top number.

34 students used: DIFF/0-*N*=*N* and MOVE/OVER/ZERO/BORROW

Whenever the top digit in a column is 0, the student writes the bottom digit in the answer; i.e., 0 - *N* = *N*. When the student needs to borrow from a column whose top digit is 0, he skips that column and borrows from the next one.

14 students used: DIFF/0-*N*=*N* and STOPS/BORROW/AT/ZERO

Whenever the top digit in a column is 0, the student writes the bottom digit in the answer; i.e., 0 - *N* = *N*. The student borrows from zero incorrectly. He does not subtract 1 from the 0 although he adds 10 correctly to the top digit of the current column.

13 students used: SMALLER/FROM/LARGER and 0-*N*=0 (203 - 98 = 205)

The student subtracts the smaller digit in each column from the larger digit regardless of which one is on top. The exception is that when the top digit is 0, a 0 is written as the answer for that column; i.e., 0-*N*=*N*.

12 students used: DIFF/0-*N*=*N* and MOVE/OVER/ZERO/BORROW

Whenever the top digit in a column is 0, the student writes 0 in the answer; i.e., 0-*N*=*N*. When the student needs to borrow from a column whose top digit is 0, he skips that column and borrows from the next one.

11 students used: BORROW/FROM/ZERO and DIFF/N-0=0

When borrowing from a column whose top digit is 0, the student writes 9, but does not continue borrowing from the column to the left of the 0. Whenever the bottom digit in a column is 0, the student writes 0 in the answer; i.e., *N*-0=0.

10 students used: DIFF/0-*N*=0 and *N*-0=0 (302 - 192 = 290)

The student writes 0 in the answer when either the top or bottom digit is 0.

10 students used: BORROW/FROM/ZERO and DIFF/0-*N*=*N*

When borrowing from a column whose top digit is 0, the student writes 9, but does not continue borrowing from the column to the left of the 0. Whenever the top digit in a column is 0, the student writes the bottom digit in the answer, i.e., 0-*N*=*N*.

10 students used: MOVE/OVER/ZERO/BORROW (304 - 75 = 139)

When the student needs to borrow from a column whose top digit is 0, he skips that column and borrows from the next one.

10 students used: DIFF/*N*-0=0 (403 - 208 = 105)

Whenever the bottom digit in a column is 0, the student writes 0 in the answer; i.e., *N*-0=0.

10 students used: DIFF/0-*N*=*N* (140 - 21 = 121)

Whenever the top digit in a column is 0, the student writes the bottom digit in the answer; i.e., 0-*N*=*N*.

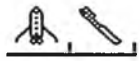
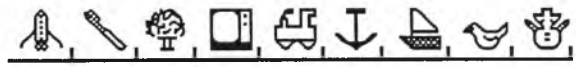
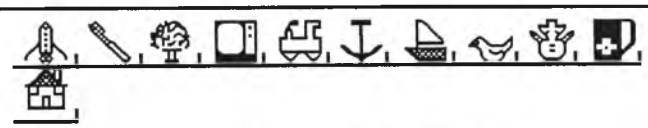
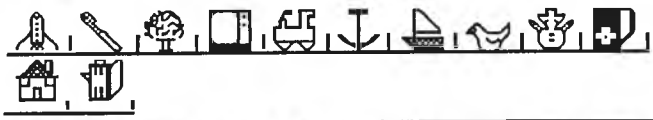
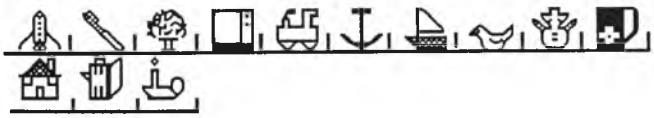
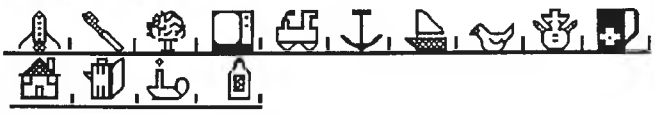
9 students used: DIFF/0-*N*=*N* and LEFT/TEN/OK (908 - 395 = 693)

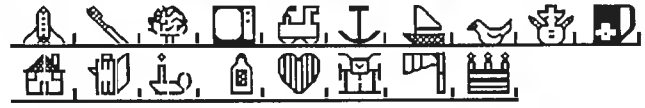
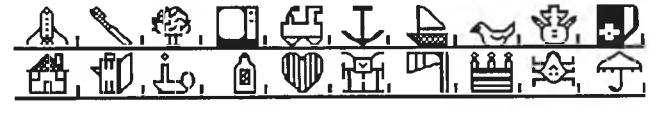
When there is a 0 on top, the student writes the bottom digit in the answer. The exception is when the 0 is part of 10 in the left columns of the top number.

APPENDIX C

The Shape Data Bank for *Number Detective* is shown in poster form on the following pages to demonstrate the range of shapes available. (In published form it is one long poster, not two separate ones as necessarily shown here.) The poster is included as part of the package to be used as a stand-alone wall chart. The "Number stories for five" section at the bottom of the poster provides an idea of what can be done using the Join procedure. Children may discover these possibilities for themselves, or they may be set as problem solving activities by the teacher or parent for children who are familiar with the program, and with the Join procedure in particular. The sample illustrated could be introduced by asking, for example, "How many different number stories for five can you make?" The child who fully understands the concept should be able to transfer the symbolic representation on screen to a numeric representation on paper, that is, 1 + 4 = 5, 2 + 3 = 5, and so on.

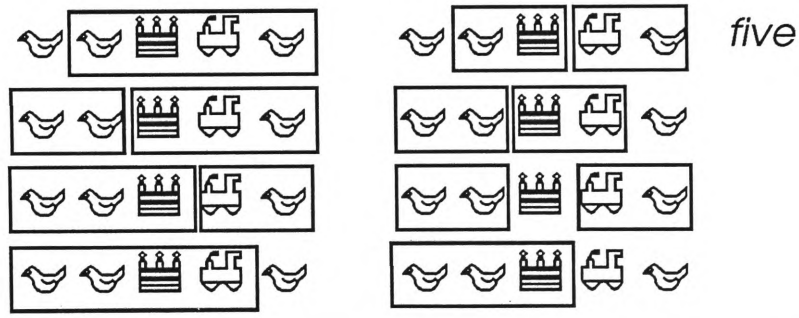
Who can count?

1		one
2		two
3		three
4		four
5		five
6		six
7		seven
8		eight
9		nine
10		ten
11		eleven
12		twelve
13		thirteen
14		fourteen

15		*fifteen*
16		*sixteen*
17		*seventeen*
18		*eighteen*
19		*nineteen*
20		*twenty*

Some Number Stories for 5

5



five